HFAQM: A HYBRID FAIR ACTIVE QUEUE MANAGEMENT MECHANISM TO IMPROVE FAIRNESS AND STABILITY FOR WIRELESS LOCAL AREA NETWORK

ATHEER FLAYH HASSAN AL-KHAMEES

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2018

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

ATHEER FLAYH HASSAN AL-KHAMEES

calon untuk Ijazah **PhD**
(candidate for the degree of)

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"HFAQM: A HYBRID FAIR ACTIVE QUEUE MANAGEMENT MECHANISM TO IMPROVE FAIRNESS AND STABILITY FOR WIRELESS LOCAL AREA NETWORK"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : ***13 Disember 2017.***
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: ***December 13, 2017.***

Pengerusi Viva: *(Chairman for VIVA)*	**Assoc. Prof. Dr. Azman Yasin**	Tandatangan *(Signature)*
Pemeriksa Luar: *(External Examiner)*	**Prof. Dr. Mohamad Yusoff Alias**	Tandatangan *(Signature)*
Pemeriksa Dalam: *(Internal Examiner)*	**Dr. Shahrudin Awang Nor**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Prof. Dr. Suhaidi Hassan**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Dr. Ahmad Suki Che Mohamed Arif**	Tandatangan *(Signature)*

Tarikh:
(Date) ***December 13, 2017***

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Pengurusan Baris-gilir Aktif (AQM) adalah satu skim proaktif yang mengawal kesesakan rangkaian dengan mengelakkannya sebelum ia berlaku. Apabila melaksanakan AQM dalam rangkaian wayarles, beberapa isu kontemporari perlu dipertimbangkan, seperti gangguan, perlanggaran, pemudaran berbilang laluan, perambatan jarak dan kesan pembayangan, yang menjejaskan kadar penghantaran pautan-pautan tersebut. Isu-isu ini mempunyai kesan langsung terhadap kesaksamaan dalam rangkaian WLAN dengan adanya jenis aliran yang berbeza. Idea utama di sebalik rangkaian wayarles ialah menggunakan fleksibiliti gelombang radio untuk memindahkan data dari titik ke titik yang memberi WLAN fleksibiliti dan kebolehgerakan: nod wayarles boleh menyambung, memutuskan atau bergerak dari satu titik akses ke titik akses lain dengan cepat. Bagaimanapun, ini memberi kesan kepada kestabilan rangkaian WLAN. Penyelidikan ini bertujuan untuk mengurangkan ketaksamaan dan ketidakstabilan dengan mencadangkan skim Saksama-Hibrid AQM (HFAQM). HFAQM terdiri daripada dua mekanisme: Mekanisme Penunjuk Kesesakan (CIM) dan Mekanisme Fungsi Kawalan (CFM). CIM telah direka untuk meningkatkan kesaksamaan dalam WLAN melalui hibridisasi penglambatan baris-gilir dengan kadar ketibaan sebagai parameter untuk mengira tahap kesesakan. Sedangkan, CFM telah dibangunkan untuk meningkatkan kestabilan rangkaian dengan menggunakan fungsi kawalan suai dengan keupayaan untuk mengugur dan menandakan paket untuk mengatasi ciri-ciri rangkaian WLAN yang cepat berubah. Satu siri kajian eksperimen telah dijalankan untuk mengesahkan mekanisme yang dicadangkan dan empat varian skim AQM; RED, REM, AVQ dan CoDel, dipilih untuk menilai prestasi HFAQM melalui penyelakuan. Penemuan menunjukkan bahawa pencapaian utama HFAQM ialah kesaksamaan 99% dan kestabilan meningkat 10% daripada skim terdekat, dengan daya pemproses lebih baik, panjang baris-gilir, kehilangan baris-gilir, dan penggunaan pautan keluar sebagai pencapaian sekunder. Skim yang dicadangkan memberikan kesaksamaan dan kestabilan yang lebih baik dalam persekitaran WLAN, dengan adanya pelbagai jenis aliran.

Kata kunci: Pengelakan kesesakan, Penilaian eksperimen, Rangkaian wayarles, Kawalan kesesakan.

Abstract

Active Queue Management (AQM) is a proactive scheme that controls network congestion by avoiding it before it happens. When implementing AQM in wireless networks, several contemporary issues must be considered, such as interference, collisions, multipath-fading, propagation distance and shadowing effects, which affect the transmission rate of the links. These issues in WLAN networks with the existence of different types of flow have a direct effect on fairness. The main idea behind the wireless network is using the flexibility of radio waves to transfer data from point to point that is giving WLAN the flexibility and mobility: wireless nodes can connect, disconnect or even move from one access point to another rapidly. However, this affects the stability of the WLAN network. This research aims to reduce unfairness and instability by proposing a Hybrid-Fair AQM (HFAQM) scheme. HFAQM comprises two mechanisms: Congestion Indicator Mechanism (CIM), and Control Function Mechanism (CFM). CIM was designed to improve fairness in WLANs by hybridizing queue delay with arrival rate as parameters to calculate the congestion level. Whereas, CFM was developed to improve network stability by using an adaptive control function with the ability to drop and mark packets to overcome the rapidly changing characteristics of WLAN network. A series of experimental studies were conducted to validate the proposed mechanisms and four variants of AQM schemes, RED, REM, AVQ and CoDel, were chosen to evaluate the performance of HFAQM through simulation. The findings show that HFAQM's main achievement is 99% fairness and improved stability by 10% from the closest scheme, with better throughput, queue length, queue loss, and outgoing link utilization as secondary achievements. The proposed scheme provides significantly better fairness and stability in WLAN environment, with the existence of different types of flow.

Keywords: Congestion avoidance; Experimental evaluation; Wireless network; Congestion control.

Acknowledgments

In the name of ALLAH, Most Gracious and Most Merciful.

"Read; And your Lord is the Most Generous, Who taught by the pen, Taught man that which he knew not;"

(The Holy Quran - Al-Alaq (96): 3-5)

At the beginning of Ph.D journey, I was always thinking about its end, but I am now thinking about its beginning. Conducting this research marks the end of interesting and eventful journey. It took a longer time to accomplish the achievement of long-waited dream than I expected. My journey has unexpected ups and downs, but it couldn't be achieved without the support of wonderful people.

I want to start by thank my supervisors Prof. Dr. Suhaidi Hassan and Dr. Ahmad Suki bin Che Mohamed Arif, for the continuous support, insight and patience during this long journey. Your constant guidance, encouragement and constructive criticism throughout all this research stages were tremendous. Thank you for allowing me to grow as a research scientist. Your advice on both research as well as on personal have been priceless.

I would like to extend my thanks and gratitude to all my colleagues at InterNetWorks Research Laboratory (IRL) for their great support and insightful comments during research camps and frequent discussions. Special thanks from the deep heart to my Ph.D. colleagues in the School of Computing, Rafid, Raaid, Hayder, Harith, Yasser and Aymen, just to name few.

Special thanks to our gurus in IRL, Dr. Adib Habbal, Dr. Mohd. Hasbullah, Assoc. Prof. Dr. Osman Ghazali, Dr. Massudi Mahmuddin, Dr. Shahrudin and others, for showing your support and providing valuable advices. You've been more than brothers to me. Further, my gratitude extended to Prof. Dr. Huda binti Hj Ibrahim (Dean of The School of Computing) and the staff and members in SOC for their guidance, assistance and kindness.

Finally, my heartiest gratitude goes to my family, to my father Flayh H. Hadi and my mother Anaam who always has faith in me, lift me when I was down and prays for my success, I hope that I can make you proud one day, to my brothers and sisters Manar, Hassanin, Yaser, Banin, Mustafa and Murtatha, who are willing to extend a helping hand. Thanks all for your love.

Declaration Associated with This Thesis

Atheer F. Hassan, Suhaidi Hassan and Suki Arif, 2015. Fairness in Active Queue Management Algorithms: An Experimental Comparison. *In proceeding of The 4th International Conference on Internet Applications, Protocols and Services* (NETAPPS2015): 135-140.

Atheer F. Hassan, Suhaidi Hassan and Suki Arif, 2017. Stability of Active Queue Management Schemes in Wireless Network: An Experimental Comparison. *Journal of Engineering and Applied Sciences*, 12: 4246-4250.

Hassan, Suhaidi, **Atheer F. Hassan**, and Suki Arif. "A Fairness Investigation on Active Queue Management Schemes in Wireless Local Area Network." *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 9.2-12 (2017): 185-195.

Hassan, Suhaidi, **Atheer F. Hassan**, and Suki Arif. HF-AQM: An Efficient Active Queue Management Scheme for Wireless Local Area Network Environment. *IEEE TENCON 2017*.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACK	-	Acknowledge Packet
AIMD	-	Additive Increase Multiplicative Decrease
AP	-	Access Point
AQM	-	Active Queue Management
ARED	-	Adaptive Random Early Detection
AVQ	-	Adaptive Virtual Queue
BDP	-	Bandwidth-Distance Product
BLACK	-	BLACK-listing unresponsive flows
CBR	-	Constant Bit Rate
CFM	-	Control Function Mechanism
CHOKe	-	CHOose and Keep
CIM	-	Congestion Indicator Mechanism
CoDel	-	Controlled Delay
CPCN	-	Congestion and Pre-Congestion Notification
DREAM	-	Deterministic with Random Ergodic Alignment Marking
DRM	-	Design Research Methodology
DS-1	-	Descriptive Study 1
ECN	-	Explicit Congestion Notification
EWMA	-	Exponential Weighted Moving Average
FIFO	-	First In First Out
FRED	-	Fair Random Early Detection
FTP	-	File Transfer Protocol
GREEN	-	Generalized Random Early Evasion Network
GUI	-	Graphical User Interface
HBF	-	High Bandwidth Flow
HFAQM	-	Hybrid-Fair AQM
HTTP	-	Hypertext Transfer Protocol
ICC	-	Internet Congestion Control
IDE	-	Integrated Development Environment
IETF	-	Internet Engineering Task Force
LBL	-	Lawrence Berkeley Laboratory
MANET	-	Mobile Ad-hoc Network
MTU	-	Maximum Transmission Unit
NED	-	Network Description Language
NS-2	-	Network Simulation Version 2
OTcl	-	Object-Oriented Tcl
PI	-	Proportional Integral
PS	-	Perspective Study
QoS	-	Quality of Service
RC	-	Research Clarification
RED	-	Random Early Detection
REM	-	Random Exponential Marking
RTT	-	Round Trip Time
SFB	-	Stochastic Fair BLUE
SHRED	-	Short-lived Flow Friendly RED
SVB	-	Stabilized Virtual Buffer
Tcl	-	Tool Command Language

TCP	-	Transmission Control Protocol
UC	-	University of California
UDP	-	User Datagram Protocol
WLAN	-	Wireless Local Area Network

CHAPTER ONE
INTRODUCTION

Congestion has long been a serious problem in Internet [1]–[4], whether the network is wired or wireless, due to their rapid growth and changes. Many mechanisms, algorithms, and protocols have been developed to overcome this issue. The Active Queue Management (AQM) algorithm is one of the significant solutions to avoid and control the congestion. This chapter presents an overview of wireless networks, followed by a brief introduction to congestion control mechanisms and algorithms. The problem statement, research questions and objectives are also stated in this chapter.

AQM has played a significant role in indicating and controlling congestion proactively in the Internet. AQM also has to work in all types of network topologies and types. With the growth of WLAN networks in recent decades, finding an AQM that works in WLAN as well as in wired networks has become urgent. Moreover, AQM has failed to achieve some objectives of wireless networks such as fairness, due to the growth of real-time Internet services and applications which require QoS; and stability, due to the rapidly changing characteristics of wireless networks.

1.1 Wireless Network

Wireless networks have increased dramatically over the last two decades and are essential in all mobile devices and laptops for access to the Internet anytime, anywhere. The main idea behind the wireless network is using the flexibility of radio waves instead of wires to transfer the data from point to point. The first wireless network standard was IEEE 802.11, published in 1997 by IEEE. Since then many standards

have been published as an improvement or enhancement to the original standard, including 802.11a, b, g and n. Each standard has its own characteristics and differs from other standards according to the band, method of encoding, or data rate used.

Wireless networks have three main components: wireless host, wireless link and base-station. The wireless host is, as in wired networks, the end-system node that runs the application or service, and uses the wireless interface to transfer the data to other wireless nodes. The base-station is part of the infrastructure of the wireless network, a node responsible for sending and receiving data from and to wireless hosts connected with the base-station through a wireless link, which is the radio wave that carries the data from one node to another. The base-station can be considered a key to communication between wireless and wired networks, also called Access Points in Ad-hoc Networks. Figure 1.1 shows the components of wireless networks [5].

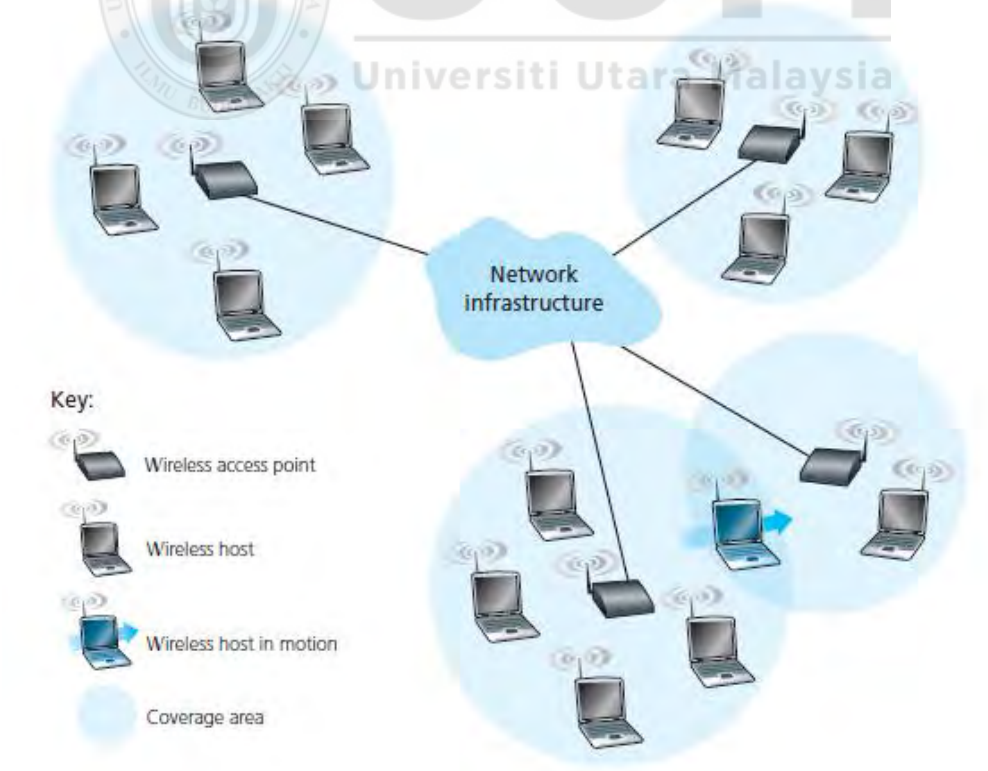

Figure 1.1. Element of a Wireless Network
Source: Kurose & Ross [5]

The key design principles behind wireless networks are flexibility and mobility. Therefore, wireless networks can be formed in many shapes and topologies. The most widely used topology that have been used for so many networks is the infratructured wireless network. This mode of the wireless network contains a base station that communicates with the wireless host from one side and with a wired network from the other side, and acting as a bridge between them. The Mobile Ad-hoc Network (MANET) is another form of wireless network which contains mobile wireless hosts that communicate between them without needing a base station. The sensor network is similar to the infrastructured wireless network but with the addition of sensors at each wireless node. The sensors sense the environment surrounding each individual wireless node and send the information to the base station.

1.1.1 Challenges

Depending on the wireless network topology, the many challenges differ from one mode to another. In general, there are many challenges that need to overcome in wireless network. Among these challenges, as mentioned by Kurose and Ross [5]:

Interference: if the sources use the same radio frequency band, interference in the frequencies will occur. For example, wireless phones using 2.4 GHz (such as the Bluetooth interface) can interfere with devices using 802.11b LANs because the 802.11b standard also uses the 2.4GHz frequency band. This interference results in neither of the devices performing particularly well. Some nodes will have full access to the access point which takes a higher share of the bandwidth, while other nodes suffer from interference through taking the lower share of the bandwidth. This results in the problem of unfairness in the access point, reflecting unfairness in AQM.

Decreasing Signal Strength: the radio waves attenuate when they pass through matter (e.g. a wall). Even in free space the signal decreases as the distance between the sender and receiver increases; this event is sometimes called path loss. Therefore, the nodes close to the access point take a higher share of the bandwidth than the more distant nodes, causing unfairness in AQM.

Multipath Propagation: this occurs when a portion of the radio signal is reflected off the surface of objects, resulting in paths of different lengths between the receiver and the sender and causing signal noise that increases the network's instability.

WLAN networks have the same problems and challenges regarding to wireless signals, with additional challenges such as dynamic environment and congestion.

Congestion: in WLAN networks, congestion occurs when many wireless nodes with high traffic load are associated with the same base station (access point), resulting in buffer overflow at the base station [6]. Many algorithms and mechanisms have been proposed to control congestion, discussed in more detail in the next section.

Dynamic Environment: flexibility and mobility are the main characteristics of WLANs, giving wireless nodes the ability to connect, disconnect or even move rapidly from one access point to another. This rapid changing will cause instability in the network and consequently queue instability [7], [8].

1.2 Congestion Control

Congestion in networks occurs when the links exceed capacity [9]. In other words, congestion occurs when the input rate at a single node interface exceeds the output rate of the link interface [10]. This results in some packets being dropped and serious delay

in transferring data; in the worst conditions the throughput drops to zero and the response time will tend to infinity [11], so there is logically no data transferred in the link. Figure 1.2 shows the simplest bottleneck topology where congestion might happen.

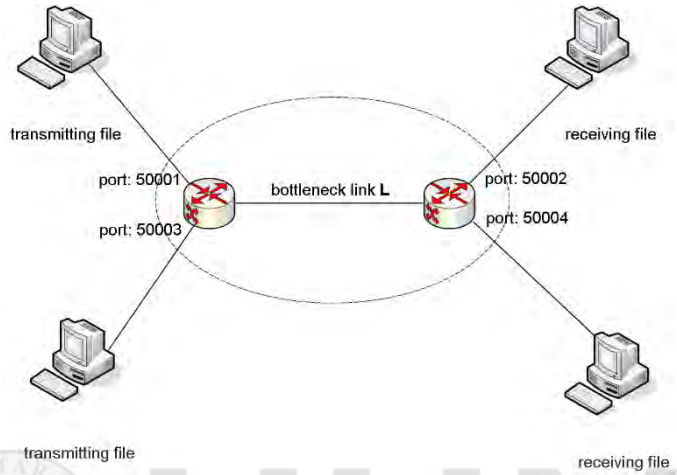

Figure 1.2. Bottleneck Topology [11]

Congestion in the Internet has been studied since the late 1980s, and many researchers have proposed algorithms, mechanisms and theories to overcome it [12]. The first congestion collapse occurred in 1986 when the data link connecting Lawrence Berkeley Laboratory (LBL) and University of California Berkeley (UC) fell from 32 Kbps to 40 bps. The terms *Congestion Control* and *Congestion Avoidance* were introduced at this time [13].

Congestion control tries to recover the network from its already congested state as soon as possible, whereas congestion avoidance tries to keep the network with as low delay and high utilization as possible, preventing congestion happening in the first place. The two methods have different algorithms, mechanisms and places where they are implemented; congestion control is mostly implemented at end-system points

(endpoint congestion control) whereas congestion avoidance is applied regularly in the middle nodes (queue management) [11]. Even when a network already has a congestion avoidance mechanism there is still a need for congestion control to recover the network if the avoidance algorithm fails. However, a congestion avoidance mechanism need not be applied if a congestion control algorithm has already been implemented, although optimal performance may not be achieved [11], [14].

1.2.1 Endpoint Congestion Control

The congestion control algorithm that designed and implemented at the end nodes (end-system point) is known as Endpoint Congestion Control Algorithms. In general, endpoint congestion control algorithms may be implemented in the application layer, although it is more usual in the transport layer of the end nodes, e.g. the Transmission Control Protocol (TCP).

There are many gaps and limitations in endpoint congestion control, the most important being that this method seeks to recover the network after congestion has already taken place, that is reactive [15]. This reactiveness can be a serious problem in networks, causing a large amount of packet loss due to the lag between the congestion drop event in the middle nodes and the source detecting the congestion loss at the end nodes. During that time the source is still transmitting at its maximum transmission rate on a link which is already congested, resulting in a very high drop rate [16], [17].

Bursty traffic is an additional insurmountable problem in endpoint congestion control. The File Transfer Protocol (FTP), web traffic and video traffic are bursty and have the same effect in all types of network (e.g. Ethernet, WLAN) [18]. In order to solve this burstiness and achieve a high level of link utilization, the middle nodes (routers) should

be implemented with large buffers. However, this solution itself causes long delays because it requires a long time in queuing, especially during the congestion. On the other hand, small buffers create another problem, which is a high packet loss, although the link delay is low. Essentially, endpoint congestion control cannot support a high level of link utilization and low queuing delay at the same time [14].

The Internet is growing quickly with the introduction of multimedia streaming and peer-to-peer applications. These applications request quality-of-service (QoS) guarantees in terms of delay rate, latency, packet loss rate and available bandwidth. This growth has magnified endpoint congestion control limitations because the current endpoint congestion control cannot fully support these demands on its own [19], [20]. It is well known that TCP damages the performance of such applications by its Additive Increase Multiplicative Decrease (AIMD) operation. Therefore, endpoint congestion control needs assistance from network resources, which need to be explored in an efficient way [21]. This is where algorithms such as Active Queue Management (AQM) and queue (or packets) scheduling have come to the rescue.

1.2.2 Active Queue Management (AQM)

There is a significant difference between *Scheduling Algorithms* and *Queue Management Algorithms* [21]–[24]. The main job of scheduling algorithms is arranging the packets in a queue, by dividing the buffer according to the number of flows to ensure that each flow has its own resource and buffer. Therefore, this type of algorithm increases fairness among different flows but it is not able to control or avoid congestion [25]. On the other hand, the queue management algorithm overcomes congestion before it happens by deciding which packet has to be dropped, when to drop it and through which port, when it has become or is becoming congested [26].

Simplicity in implementation is one of the properties of the queue management algorithm that make the implementation as simple as applying First-In-First-Out (FIFO) queuing for all the flows, or maintaining the pre-flow state. The scheduling algorithm, however, requires complex implementation for the resource allocation process [21]. Thus, this research focuses on AQM.

The droptail queue is one of the earliest queue management algorithms, adopting the FIFO operation. It was initially the only queue management algorithms implemented on the Internet [15] due to its simplicity and easy application in the routers. However, droptail queues increase the limitations of the endpoint congestion control algorithm (TCP) as its FIFO mechanism causes a "lock-out", meaning that a small number of flows will share link resources such as buffers and link bandwidth, blocking other flows [15], [18].

The droptail algorithm simply drops all incoming packets in reaction to a full buffer. This drop notifies the TCP sources of the congestion when the overflow has already occurred, reducing the sources' transmission rate and congestion windows that will lead to network instability, high delay and very low throughput [27]. Therefore, the process of recovering the queue from its overflow state to the underflow state involves a noticeable time delay, leading to queuing-delay variance known as *jitter*. In addition, the droptail algorithm suffers from long queue delay especially when large buffers are filled up. This delay, together with jitter, makes real-time applications impractical. TCP itself also has a clocking mechanism as the time variance between sending packets and acknowledgment packets (ACK) causes the burstiness [28].

The droptail algorithm penalizes the TCP flows as the dropping policy increases the Round-Trip-Time (RTT). The TCP sending rate (hence window size) is inversely related to RTT [29]. TCP flows with long RTT will be slower than those with a short RTT to increase the sending rate and therefore the throughput by taking lesser slots in the queue, as shown in Figure 1.3. However, in the droptail queue all TCP flows will suffer from packet drop regardless of how many slots in the queue a single flow occupies. Thus, the flows with large RTT suffer more in the droptail policy [30]. Accordingly, AQM algorithms were designed.

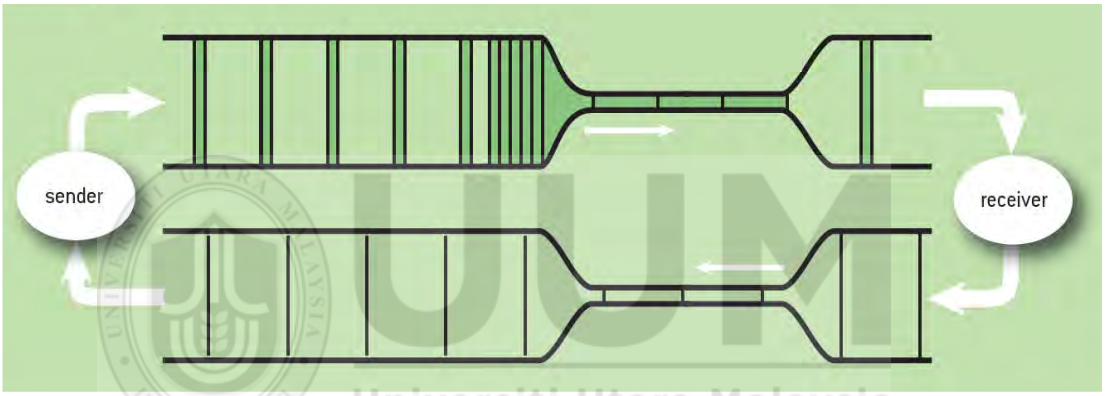

Figure 1.3. TCP connection after one RTT
Source*:* Nichols & Jacobson [31]

AQM, unlike the droptail queue, is a proactive congestion avoidance algorithm [8]. In general, AQM has three components: (1) congestion indicator, (2) control function, and (3) feedback mechanism [32]–[35]. The congestion indicator detects when the congestion occurs or is close, and the control function decides what has to be done when congestion has been indicated; the feedback mechanism is the signal that will be sent to notify the sources about the congestion in order to reduce their sending rates. These three mechanisms are what make AQM proactive. Figure 1.4 shows AQM's components.

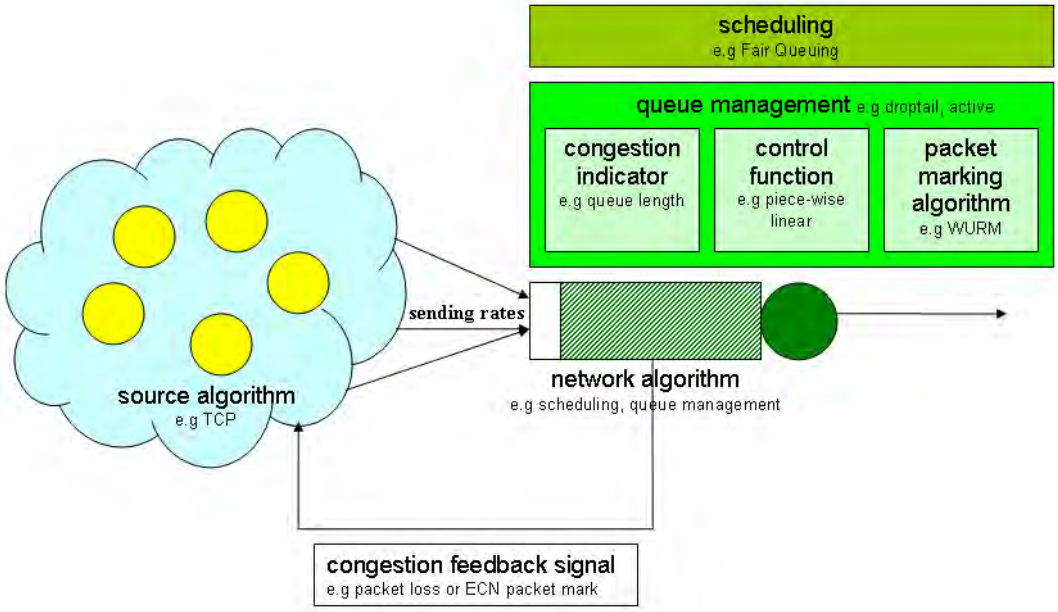

Figure 1.4. AQM components
Source: Adams [36]

The congestion indicator is the mechanism that anticipates congestion by monitoring the incoming rates (rate-based AQM) and indirectly keeps the queue length to a target value by controlling the input rates [18], or by monitoring the queue length itself (queue-based AQM) and keeping it below a targeted value [37]. Some schemes use both rate-based and queue-based techniques together to detect congestion so as to control the incoming rate and the queue length at the same time. The main goal of the congestion indicator, therefore, is to keep the queue length at its targeted value.

The control function calculates the probability of marking or dropping of incoming packets according to the congestion level as given by the congestion indicator [38]. Its main task is to determine which packet must be marked or dropped. This function is regularly ignored by researchers, but it has given good results in terms of packet loss and delay in many algorithms [27]. It works side by side with the feedback mechanism. The feedback mechanism uses the Explicit Congestion Notification (ECN) marks to

signal to the source nodes to maintain their sending rates. It has been proved that ECN-marking is better than dropping packets in terms of stability and latency [39].

1.2.2.1 AQM Design Approaches

The working principle of AQM mechanisms and components have been introduced, but an understanding of AQM component design is crucial at this point. This section describes the main approaches in implementing each component of the AQM.

The early congestion indicators were based on a simple mathematical equation in the rate-based mode, comparing the incoming rates by subtracting them from the manually configured targeted values, as in the queue-based method. Modern indicators have derived the formulation from the early one or have introduced new proven ones, but in general they use a mathematical equation or equations that calculate the congestion level [40].

The control functions of most AQM algorithms are clearly classified into three main approaches: *heuristic*, *theoretic* and *optimization*. The main purpose behind the heuristic approach is to make the control function stand on its own, intuitively tuning the function parameters automatically according to historical events [41] that performed well, especially when compared with droptail queuing. However, even AQMs that have been designed based on the heuristic approach still need manual configuration of some parameters, depending on network conditions [7].

The principle behind the theoretic approach is that AQM has to work with endpoint congestion control algorithms simultaneously, so that both endpoint congestion control and the AQM control function are modelled. The model for endpoint

congestion control is designed and developed to capture some TCP AIMD process information (the congestion avoidance phase) and ignore others, such as slow start and retransmission timeout [42]. These ignored parameters have an important impact on the behavior of short TCP flows, which is a serious limitation in TCP modelling [20].

Optimization is the third design approach; it formulates the control function problems mathematically and tries to maximize link utilization either by optimizing the source sending rates or by optimizing the congestion measures [35]. Therefore, this approach has two concepts. First, the AQM decides how much link capacity is available for the source sending rates, and the network's maintaining these rates. Secondly, the sources decide how much capacity is needed for the sending rates, and the AQM allocates this capacity accordingly [43]. This severely limits this approach in terms of fairness [36].

1.3 Research Motivation

The rapid growth of the Internet has increased the demand for algorithms and protocols to increase network performance. AQM is especially important for its proactiveness to control congestion. Its main objective is detecting and avoiding congestion, with low queuing delay and high link utilization; the AQM also has to achieve other goals, such as stability, fairness, scalability, robustness and responsiveness.

In terms of stability, AQM should react efficiently when network conditions change suddenly. In other words, AQM must provide stable performance when the number of connections increase unexpectedly, avoiding queuing delay or keeping it as low as possible, since this network's condition is certain to cause queue overflow and under-utilization. With the growth of real-time applications, it is essential to consider stability in AQM performance.

Fairness was one of the earliest goals of AQM. With variation in Internet flows, it is very important to provide equal queue sharing among all the flows regardless of the individual flow type: whether it is a bursty (web traffic), unresponsive (UDP) or responsive (TCP) network connection. AQM algorithms try to approximate the value of fairness for the flow aggregations by dropping packets appropriately so as to limit the congestion and allocate a suitable length for the queue [21], [22]. The AQM under optimization approach does not achieve the optimal state in terms of fairness even if it guarantees QoS among different flows. Furthermore, optimization approach has a very big gap in terms of stability especially in wireless networks. In addition to all the above, the Internet Engineering Task Force (IETF) is encouraging researchers to develop and improve AQM algorithms [18], [27], [44].

Since 1998, RFC 2309 [45] has been ratified by the IETF community which outlines recommendation on queue management and congestion avoidance in the Internet. It is strongly recommended that AQM should avoid the lock-out phenomena which happens in the router when few flows monopolize all the queue space, preventing other connections from getting room in the queue. This phenomenon is the main reasons for the unfairness. For decades AQM has been an elegant and a promising technology that have been taken extensive discussion and debate in IETF meetings until the last RFC 7567 that stated the last IETF recommendation regarding AQM. RFC 7567 [46] clearly states the presence of lock-out issue and has suggested researchers to investigate deeply in the issue of unfairness. During the past ten years, 32 IETF meetings have been held. Although, AQM has appeared in IETF meeting's agenda for eight times, before 2013 similar topic have been discussed many times with different terminologies such as Congestion and Pre-Congestion Notification (CPCN) which has appeared 13 times. On the other hand, topics like Internet Congestion Control (ICC) has appeared

20 times for the past ten years [18], [27], [44]. Figure 1.5 shows the percentage of each topic that appeared in IETF agendas for the past ten years.

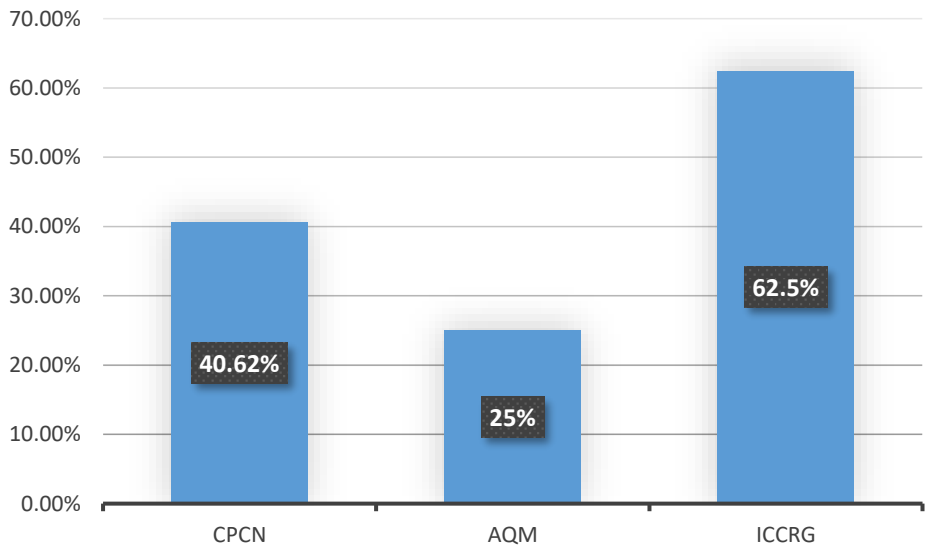

Figure 1.5. The percentage of the topic AQM in IETF agendas for past ten years.

1.4 Problem Statement

The earliest AQM algorithms were designed to overcome issues in wired networks, such as congestion avoidance and control, which are still an open issue. However, the growth of wireless technology has raised other issues, such as the heterogeneity of different network technologies with different characteristics and properties. When implementing AQM in wireless networks several new issues have to be considered, such as interference, collisions, multipath-fading, propagation distance, shadowing effects and mobility, regardless to the wireless network type (infrastructured or non-infrastructured) [6], [47]. Therefore, there is an urgent need for an AQM algorithm that can perform in wireless networks as well and as efficient as in wired networks [15], [48], [49].

Fairness and stability are among the important objectives of AQM algorithms. Achieving such objectives requires accurate design of the main three components of AQM algorithms: congestion indicator, control function and feedback mechanism. Therefore, many approaches have been used to design AQM algorithms [50], [51].

The deterministic optimization approach is the part of the optimization approach that considers designing AQM to perform effectively in WLAN networks [36]. It involves three different hybrid algorithms: Random Early Marking (REM), Stabilized Virtual Buffer (SVB), and Adaptive Virtual Queue (AVQ) that tried to sustain the issues of the wireless networks [8], [35], [39]. Although this design method significantly improved the performance of wired networks there are still problems with wireless networks, especially in terms of stability [36].

Fairness is providing all flows with an equal amount of shared link. In order to achieve fairness between different types of flow in AQM schemes, the congestion indicator should be designed not only to provide fairness but also to calculate the congestion level accurately. It can then indicate flows that exceed their share from the link. Early congestion indicators used different types of parameters to calculate congestion, such as queue-based, rate-based, load-based, loss-based and a mixture. These types of indicator have not filled the fairness gap in AQM, especially in wireless networks [52].

In WLAN networks, the wireless nodes can connect, disconnect or even move from one access point to another abruptly. This rapid change causes instability in the network. However, achieving stability in buffer will partially overcome instability at the network level, since the control function in AQM is responsible for dropping or marking packets, which controls the queue length and queue drops in the buffer,

having a direct effect on the stability [7], [8]. However, the control function should be adaptive to rapid network changes if it is to achieve stability. Therefore, this research is proposing a hybrid AQM algorithm, HFAQM, that takes the advantages of the optimization approach and fills the gap in stability and fairness in WLAN network environments.

1.5 Research Questions

The following questions were raised when addressing AQM performance problems in terms of fairness and stability in WLAN network environments:

i. How to enhance fairness among different types of flow?

ii. How to improve stability in the WLAN network environment?

iii. What is the impact of the proposed scheme on the WLAN environment?

1.6 Research Objectives

The main aim of this research is to design a new hybrid AQM algorithm that improves fairness and adds more stability in wireless network environments. The following specific objectives have been outlined in order to achieve the aim:

i. To design a hybrid congestion indicator mechanism to enhance fairness among different types of flow.

ii. To design an adaptive control function mechanism with a hybrid feedback mechanism to improve the stability in WLAN network environments.

iii. To integrate and implement the proposed mechanisms to improve fairness and stability in WLAN environments.

1.7 Research Scope

The overall goal of this research is to design a Hybrid Fair AQM (HFAQM) algorithm for WLAN networks. Specifically, HFAQM is proposed to provide fairness among different types of flow and stability in the wireless environment with the ability to detect and avoid congestion. It can be used for any WLAN network system that requires fairness and stability with any kind of application.

1.8 Significance of the Research

This research is introducing a new HFAQM algorithm that can detect and avoid congestion. Furthermore, the proposed algorithm is capable of providing fairness among responsive and nonresponsive flows with more stability in WLAN networks. Finally, HFAQM might be able to provide a high link utilization with multiple flows in a fair and stable manner with low queuing loss in wireless routers.

1.9 Organization of the Thesis

This thesis consists of six chapters, which are organized as the following:

Chapter One: This chapter gives an overview about the area of the research, which also includes research motivation and introduces the problem statement. This chapter also addresses the research questions and research objectives, presents the scope, the steps, the contribution and the significance of the study.

Chapter Two: This chapter covers the literature review from the previous and current related work. Moreover, this chapter presents relevant information for better understanding of the selected research area.

Chapter Three: This chapter provides the details of the selected methodology that is used for this study.

Chapter Four: This chapter presents the Congestion Indicator Mechanism and Control Function Mechanism. In addition, it addresses the approach the proposed mechanisms are based on. Moreover, a verification and validation for the introduced mechanisms are provided.

Chapter Five: This chapter presents HFAQM Scheme. The integration of congestion indicator and control function mechanisms in order to obtain high performance is explained. In addition, an extensive evaluation of the final scheme is introduced with results discussion.

Chapter Six: This chapter concludes the study, highlights the contributions in the body of knowledge, recommends for the future work and shows the limitation of this study.

CHAPTER TWO
LITERATURE REVIEW

This chapter offers a comprehensive discussion of AQM in relation to the research goals, with greater detail of fairness and stability in AQM. The factors that affect AQM performance and the classification of AQM are reviewed. The chapter then focuses on AQM algorithms that have improved fairness and stability in one aspect or another. AQM design is based on many theories, some of them directly related to computer science, and others introduced for other fields.

2.1 Active Queue Management (AQM)

AQM has played a significant role in the Internet since 1993 when the first AQM was proposed by Jacobson [48]. It was presented to complete a TCP protocol to avoid and control congestion. Further advantages of AQM are fewer dropped packets, low end-to-end latency and avoiding lock-out behavior by keeping the queue size as low as possible, as stated in RFC2309. With the rapid growth of the Internet, AQM must satisfy other network requirements such as robustness, responsiveness, fairness, stability and scalability. Therefore, a large number of AQM schemes have been proposed to meet these additional Internet requirements (e.g. QoS). Random Early Detection (RED) was the first formal AQM proposed by [48], followed by many schemes, some of which were only improvements of the RED algorithm itself. Other AQM schemes can be considered as completely new algorithms. The best way to study such a large number of schemes is to classify them into groups by their characteristics.

There are many ways to classify AQM schemes due to AQMs' characteristic and how different from one to another. There are three main classification schemes have been

proposed in the literature for comparison and study: based on their mechanisms (e.g. types of congestion indicator), context of use (e.g. wired or wireless networks) or performance (such as throughput, packet loss, fairness, etc.).

A. Classification by Mechanism

AQM mechanisms can be divided into five different groups which are congestion indicator, control function, parameter tuning, flow differentiation, and feedback signal; as can be seen in Table 2.1. AQM schemes can be classified by the mechanisms used in the algorithm or by a combination of mechanisms based on the characteristics of AQM itself. For example, schemes classified by their indicator can be divided into queue-based, rate-based, load-based, packet-loss-based, or a hybrid of any two of these.

AQM schemes have many parameters in their design. In early AQMs, parameters had to be configured manually according to the network system characteristics that AQM will be implemented. Later schemes were proposed to maintain their parameters dynamically, based on network load, bandwidth-distance product (BDP), RTT, or link capacity. Additionally, flow type is an especially important consideration in AQM design, as non-responsive flows and short flows can penalize responsive flows. Therefore, some schemes have a built-in mechanism to differentiate between flow types, used as flow aggregation or pre-flow effectiveness.

Control function mechanisms can be divided into three categories: heuristic, theoretic and optimization. Many plausible AQM schemes with sophisticated control algorithms have been mooted as linear, non-linear, optimal and robust control, while others address delay compensation. However, the main point for theoretic approach focuses

on responsive flows, usually in TCP models which have congestion avoidance mechanisms in their own design. On the other hand, optimization approaches have been explored widely and comprehensively under deterministic and neural networks optimization, but no algorithms have been proposed for stochastic optimization.

Table 2.1
AQM Scheme Classified by Mechanisms

AQM Mechanisms	Mechanism Type	Characteristics	AQM Schemes
Congestion Indicator	Queue-based	Enqueue-events only	RED, GRED, CHOKe, DRED
		Enqueue & Dequeue events	FRED
	Rate-based	Arrival rate	LUBA, SFED, RARED
		Congestion window size	SHRED
	Load-based	Flow Count	GREEN, BLACK, SFED
		Traffic composition	RED Worcester
	Packet Loss	Loss Volume	LRED
		No. of buffer over/under flow events	BLUE
	Mixture		Load/Delay Controllers, Yellow
Parameter Tuning	Static		RED, GRED
	Dynamic	Network load	GREEN
		Bandwidth-Distance Product (BDP)	---
		RTT	GREEN
		Link Capacity	GREEN
	Mixture		ARED, A-RIO, PSAND
Flow Differentiation	None		RED, GRED
	Flow aggregates	Two-class system: unresponsive vs TCP	SFB, RIO-C
		Two-class system: Web vs FTP	SHRED
		Multiple-class system	RIO-CD, WRED, Rb-RIO, D-CBT, SFED
	Individual flows		FRED, BRED, BLACK
Control Function	Heuristic	Equation-based	RED, GRED, DSRED
		Non-Equation based	MRED
	Control Theoretic	Classical and Modern control	PI, PD, PID, P-PID, PI-PD, PD-PD
		Fuzzy Control	Fuzzy Control RED, Adaptive Fuzzy RED, FUZREM, Fuzzy GREEN, Deep BLUE, Fuzzy PID, Fuzzy CHOKe
	Optimization	Deterministic	REM, AVQ, SVB
		Neural Networks	Neural-Network-based PID controller
Feedback Signal	Packet dropping	Random	RED, GRED, CHOKe, REM, SVB
		Deterministic	AVQ
	ECN Marking	Random	REM
		Deterministic	AVQ

Adopted from [36]

Finally, AQM algorithms can be differentiated by feedback mechanisms, as well, which can be easily classified as implicit packet dropping or explicit ECN packet marking, whether random or deterministic. Table 2.1 shows the AQM classification by mechanisms and highlights well known schemes for each mechanism or approach.

B. Classification by Context of Use

The ideal AQM scheme should perform efficiently regardless of the context of use. For example, AQM should give the same performance whether implemented in a wired or a wireless network. However, different contexts may have different requirements and characteristics that might require a specific AQM configuration. Figure 2.1 presents the classification by context of use. The main baseline for differentiation is whether the AQM was designed to support maximum link utilization (best-effort network philosophy) or to consider and improve a differentiated-services network paradigm to provide QoS guarantees. Under these two categories, the AQM schemes tend to be designed and implemented specifically for wired or wireless networks, as these have very different characteristics and properties. Wireless networks can be sub-divided into ad hoc or infrastructure types.

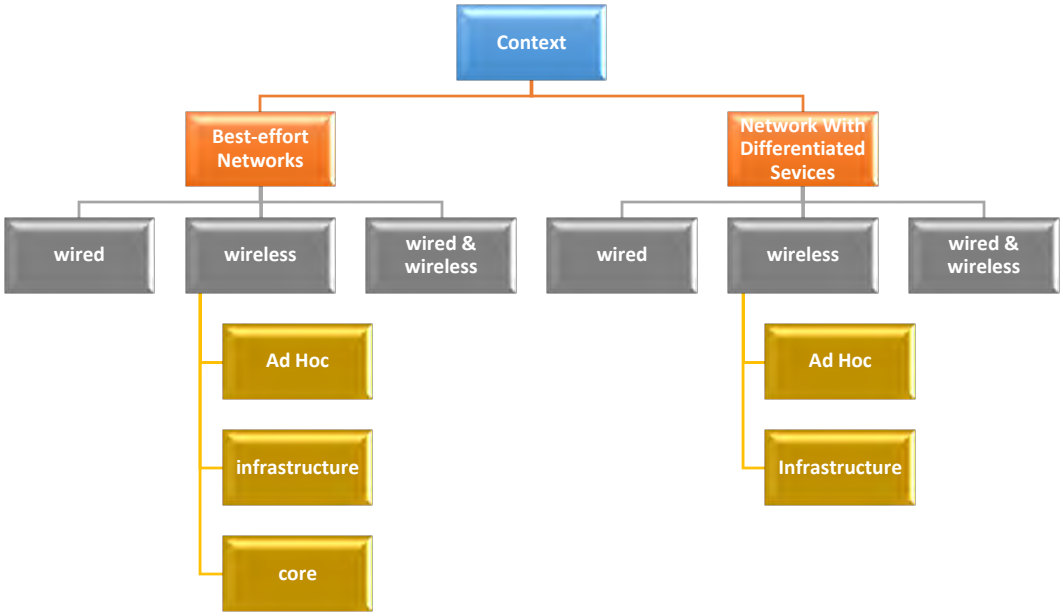

Figure 2.1. AQM classification by context of use

C. Classification by Performance Criteria

Figure 2.2 shows the performance criteria used to compare AQM schemes, into three categories: steady-state, transient and complex behaviour. The figure shows that fairness and queue stability are in different groups, so that steady-state and transient behaviour should both be considered in analyzing fairness and stability in AQM algorithms.

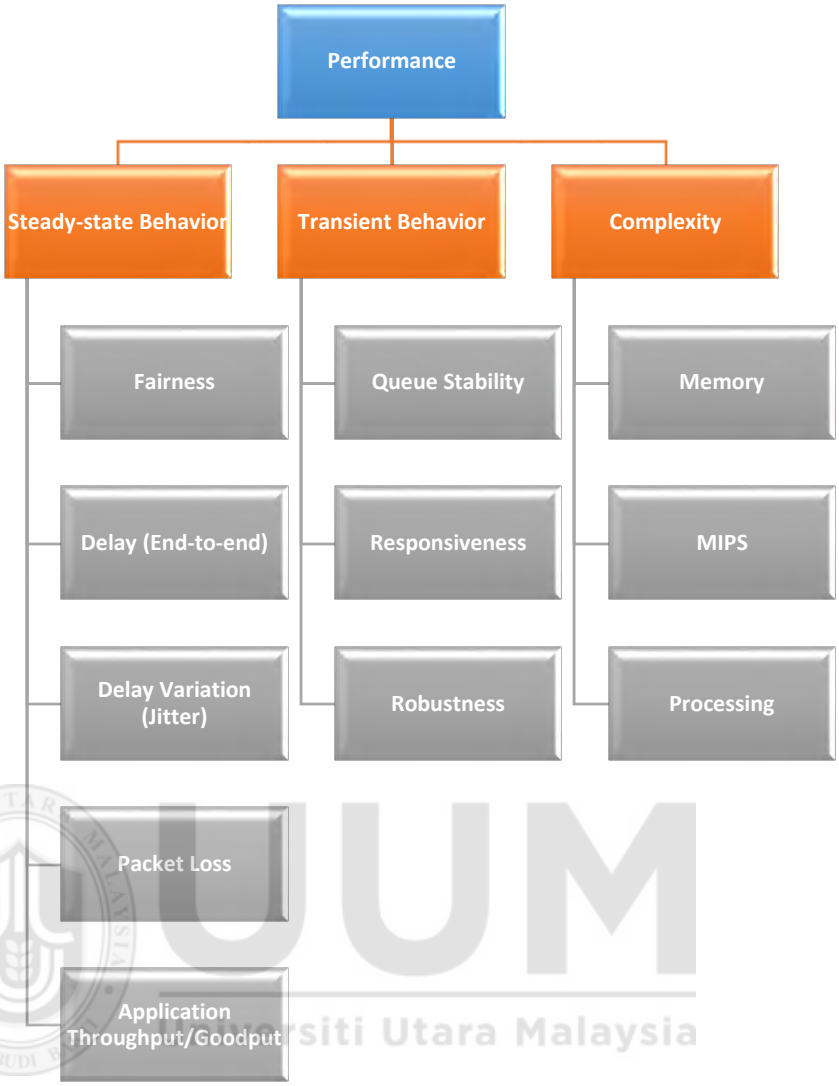

Figure 2.2. AQM classification by performance criteria

Classification by mechanism is considered the most efficient method of categorizing AQM schemes, and has been used in most of the literature researches for its accuracy. Besides that, some researchers have used classification by context of use or performance criteria, and others have proposed their own classification methods to meet their own research characteristics. Given the objectives of this research, the literature on AQM algorithms that improve fairness and stability will be studied in this research literature review and because the research focuses on three mechanisms,

congestion indicator, control function and feedback, other schemes will be referred to as appropriate.

2.1.1 The Ideal AQM

As already established, AQM should achieve high network stability, scalability, robustness, fairness and responsiveness in addition to the main goals of congestion avoidance, congestion control, low queueing delay and high link utilization. Therefore, designing an ideal AQM scheme should take all these goals into consideration, although achieving all of them is difficult. In order to motivate researchers to design an optimal AQM scheme, many points have been highlighted as design considerations. The ideal AQM scheme should ensure the following characteristics [18], [53]–[55]:

i. Burst flows: AQM should be able to deal with traffic burstiness to reduce unnecessary packet drops.

ii. Reduce jitter: it is very important to reduce the time between arriving at and leaving the queue for each packet and to keep the queue at an average level because empty flows reduce link utilization.

iii. Delay: working with large delay networks is important as well.

iv. RTT: AQM should deal with different RTTs without adding more bias because TCP already controls this.

v. Multiple bottlenecks: AQM should deal with multiple bottlenecks environments without penalizing the flows that have already been controlled by other nodes.

vi. Congestion notification: AQM should send notifications to sources at a suitable rate to keep input rate below output rate.

vii. Adaptation to change: AQM should be able to control the changes in the nature and amount of flows.

According to [26], AQM will not be able to work optimally in all scenarios because different environments need different requirements which lead to different parameters in configurating the AQM algorithm. Therefore, AQM schemes will accomplish better stability as a consequence of responsiveness; low latency at a consequence of ideal throughput; and fairness at a consequence of simplicity.

2.1.2 Factors that affect AQM performance

There are many factors that can affect AQM performance, including the following as highlighted in [53].

i. Queue size and link capacity: these factors are pre-determined; they have to be time-invariant.

ii. RTT time and traffic load: AQM should be sufficiently robust to deal with dynamic changes on traffic load and RTT because these will affect network stability.

iii. Flow synchronization: many TCP flows all in-synch with each and other sharing same media will increase burstiness in the network and affect AQM performance [27].

iv. Routing matrix gain: there is a relationship between robustness and routing

[56]. Routers use a routing matrix gain to represent the topology mathematically, affecting congestion and hence AQM performance.

v. Flow variants: short-lived and unresponsive flows together with TCP long-lived flows (and even a mixture of TCP versions) affect AQM performance.

vi. Reverse-path asymmetry: congestion in a reversed path causes ACK packet loss and that will increase burstiness in TCP [14].

2.1.3 AQM Fairness and Stability

Internet traffic nowadays is mostly carried by TCP protocols (including HTTP, FTP and TELNET) which has shown good improvement in congestion control due to its AIMD strategy. However, the number of streaming applications is growing dramatically, and the UDP protocol is the major protocol that have been used in these applications, providing large volumes of traffic with higher transfer rates and causing congestion collapse [13], [57]. This type of unresponsive protocol can penalize other responsive protocols, consume an unfair amount of the bandwidth, and cause system instability.

Fairness is one of the earlier goals of AQM, its main concept being to provide equal shares of a queue among different types of flow. However, it is hard for AQM to differentiate between flows without supporting information to define the flow type, whether responsive, unresponsive or short. Therefore, AQMs with per-flow information give better results than without per-flow information AQM. Nevertheless, keeping the buffer occupancy equal for each individual flow does not mean the output rate from the buffer will be equal [22], [58], [59]. The recommended way to guarantee

fairness among multiple flows is by combining scheduling algorithms with AQM algorithms [25], [29], [59], [60]. However, the main difficulty with this combination is a conflict between AQM objectives and scheduling objectives, since the former aim to keep the queue as short as possible, and the latter require longer queues for greater efficiency [60]. Rate-based AQM algorithms have a stronger effect on fairness and QoS than scheduling algorithms [29]. Many AQMs have been proposed with respect to fairness such as Fair RED (FRED) [30], Short-lived Flow Friendly RED (SHRED) [61], CHOKe [22], GREEN [62], Stochastic Fair BLUE (SFB) [63], and BLACK [64].

Stability has been among the important objective of AQM. Stable queues have played a great role in designing AQM scheme due to its ability for maintaining higher throughput and lower delay, which has a significant effect in many Internet applications. However, designing an AQM able to ensure a stable queue size in different network conditions is not an easy task. As already mentioned, a large queue size increases delay in the network, the main reason for network instability. Stability and short queue size are therefore related and connected [65], [66]. In addition, a network with a large propagation delay will affect system stability [66], [67]. There are other factors that penalize network stability include low congestion level and low target queue size [66]. According to [43], [52], [68]–[70], the optimization approach shows remarkable results in terms of stability, and it has been proved mathematically that AQM schemes designed with the deterministic optimization approach has much improvement in terms of stability, effectiveness, robustness and fairness. The most important AQM schemes reported in the literature using this approach are Adaptive Virtual Queue (AVQ) [8], Random Early Marking (REM) [35], and Stabilized Virtual Buffer (SVB) [39].

2.2 Fairness and Stability AQM Schemes

There are many schemes and algorithms that have been proposed guaranteeing fairness and stability, some have their own novel mechanisms and others are just improvements of previous algorithms. Each has its own advantages and drawbacks. This section surveys some of these whose impact is strongest in the fairness domain (RED, FRED, Stochastic Fair BLUE, and BLACK), the stability domain (AVQ, REM, SVB), or both domains together (CoDel).

2.2.1 Random Early Detection (RED)

RED was the first formal AQM proposed to replace the droptail queue algorithm in TCP/IP networks [15], [19], [26], [32], [48]. It is a queue-based AQM with no per-flow information that provides the network with a congestion avoidance mechanism. It can be considered under heuristic design, which uses statistical probability packet dropping when the queue length reaches a specific threshold value [28], [30], [71]. Besides congestion avoidance and control, RED has been designed to achieve fairness among different bursty flows [8], [54], minimize queueing delay by controlling the queue lengths in low values [8], [15], [72], prevent the interconnection between global synchronization and packet dropping [73]–[75], reduce packet loss, and achieve high link utilization [15]. RED is considered in detail here, for its importance in this research study and because it was the foundation in the design of many newer AQM schemes and is the most studied algorithm in AQM research [37], [44], [76].

RED uses Exponential Weighted Moving Average (EWMA) [77]–[80] for the queue length as a congestion indicator and for calculating the congestion level at the queue. This average is updated with every packet arrival and estimates the actual queue length [73]. It can be calculated as:

$$\bar{q}(t_{i+1}) = (1 - w_q)\bar{q}(t_i) + w_q q(t_{i+1}) \quad (2.1)$$

where

$q(t)$: $instantaneous\ queue\ length\ at\ time\ t$

$\bar{q}(t)$: $average\ queue\ length\ at\ time\ t$

w_q: $(EWMA)\ queue\ weight$

t_i: $arrival\ time\ of\ the\ i^{th}\ packet$

EWMA (w_q) is a static parameter that must be configured accurately. Besides, RED has three further parameters used for the control function: minimum threshold min_{th}, maximum threshold max_{th}, and maximum non-congestion probability P_{max}. If the average queue length is below min_{th} then RED will work normally without any changes, but if it increased to a value between min_{th} and max_{th} it starts to drop incoming packets, constrained by the proportional probability function to reduce average queue length. When the average queue length increases above max_{th} RED drops all incoming packets without exception, as shown in Figure 2.3 [39], [81], [82]. The RED control function can be expressed mathematically as:

$$p(\bar{q}) = \begin{cases} 0, & 0 \le \bar{q} \le min_{th} \\ \frac{\bar{q} - min_{th}}{max_{th} - min_{th}} P_{max}, & min_{th} \le \bar{q} \le max_{th} \\ 1, & \bar{q} \ge max_{th} \end{cases} \quad (2.2)$$

where

p: $packet\ dropping\ probability$

P_{max}: $maximum\ uncongestion\ probability\ of\ dropping\ at\ the\ max_{th}$

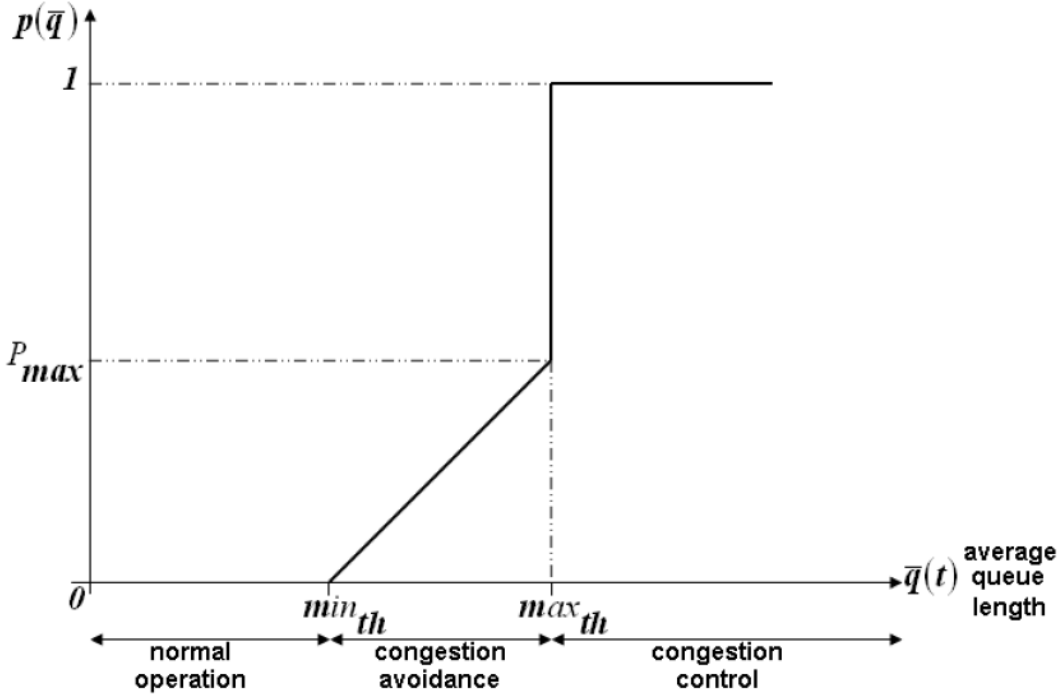

Figure 2.3. RED Control Function
Source: Adams [36]

Despite its importance, RED has many drawbacks. One of its major problems is the parameter tuning [76], [83], [84], as it has four important parameters which are very sensitive and depend on network conditions. Therefore, a set of parameter values might work perfectly with a certain network load and delay, but imperfectly with the next load and delay, which is likely to occur because of the Internet's rapidly changing characteristics [7], [11], [85], [86]. Another significant problem with RED is using queue length as both a performance measure and congestion indicator at the same time; this coupling results in deterioration of throughput and delay with increasing traffic load [84], [87], [88]. According to [41], when the number of flows increases the marking probability should increase, but in RED this means the queue length should also be increased. However, it has a fixed value regardless of increased flow, which leads to instability. Similarly, RED does not differentiate between different types of flow or TCP flows with different RTT, which again penalizes the stability. The packet

dropping probability does not guarantee fair bandwidth share among the flows [30] because RED drops all incoming packets with the same probability regardless of the number or type of flows. The result is higher packet loss whatever the rate or type of flow, even if one has not reached its fair share of the bandwidth.

2.2.2 RED variants and related schemes

RED has been developed in many new AQM algorithms not only because it was the first AQM proposed for the TCP/IP networks but also to overcome its inability to solve such problems as parameter tuning, fairness, stability and flow differentiation, as listed in Table 2.2. Algorithms proposed to enhance RED in terms of fairness and stability include FRED, SFB and BLACK, which all have some significant improvements but also some drawbacks.

Fair RED (FRED) is RED with a per-flow information state, developed to overcome RED's unfairness problem [28], [81]. FRED has two additional parameters, min_q and max_q, to represent the minimum and maximum number of packets allowed to enter the queue for each flow. If the flow's buffer occupancy (represented by q_{len}) is less than min_q and the average queue length is below max_{th}, the flow will not suffer from any loss. A flow will experience packet drop when its buffer occupancy q_{len} exceeds max_q or the average queue length becomes greater than max_{th} which is the same as RED's dropping policy. min_q is adjusted dynamically with the global variable *avgcq* which is the average per-flow queue length and can be calculated by dividing the average queue length by the number of flows. FRED counts how many times max_q has been exceeded and keeps the value in a parameter called *strike* for each flow. The flow is not allowed to exceed its *avgcq* when it has a higher strike value than other flows, thus penalizing unresponsive flows from using up most of the queue space [30], [59].

Table 2.2

Summary of some RED's drawbacks and resultant AQM schemes

RED Shortcomings	Proposed Solution	Resultant AQM
Congestion Indicator		
EWMA queue length unable to detect incipient congestion due to short-term traffic load	Use other congestion indicators apart from or in coordination with EWMA queue length	
EWMA queue length can induce RED to drop packets unnecessarily	Use instantaneous queue length. However, in conjunction with other congestion indicators	SRED, LRED, MRED
EWMA reduces the responsiveness and stability of RED in high bandwidth-distance products and bursty flows		
Small w_q can cause instability in queue length		
w_q affects fairness index for low and medium loads		
Sampling upon every packet arrival is unnecessary	Provide a fixed sampling interval	Proportional Integral (PI)
Sampling that only considers packet arrivals and not dequeue events causes average queue length to deviate far from true length	Include dequeue events in congestion indicator	FRED
Direct coupling of queue-length as a congestion indicator and performance parameter	Provide a fixed queue length for predictive queueing delay	PI, PID, DRED
Reduction in source rates not profound		
Control Function		
Effective RED has a range of reference queue length between min_{th} and max_{th} instead of a single target	Specify a single reference queue-length	Control-theoretic approaches: PI, PID, etc. DRED, ARED
The difference between min_{th} and max_{th} determines the size of oscillation around an equilibrium point		
If min_{th} is too close to max_{th}, RED will lead to global synchronization		
If max_{th} is too low, RED essentially acts as droptail		
Non-adaptive, manual, static parameter tuning	Make parameters adaptive according to changing network conditions	ARED, GREEN, BRED
No load information associated with thresholds		
If *Pmax* is too small and *N* is large, there is be insufficient congestion feedback		
The larger the slop of RED's control function, the faster the convergence but less stability		
The discontinuity causes oscillations in queue length	Remove discontinuity	GRED
Proportional packet dropping does not guarantee fair share of bandwidth	Add mechanisms to improve fairness	FR, SHRED, BRED, BLACK, SFB
Unresponsive flows drive up drop rates for all flows		
Feedback signal		
For a given dropping probability, the uniformity of packet drops may not be realized	Improve the spacing between packet drops	DREAM

Adopted from [36]

FRED suffers from several drawbacks. According to [59], [89], it cannot guarantee fairness in most cases and scenarios, although it is fairer than RED. On the other hand, FRED has a limited number of flows because the queue size is limited, and needs a large buffer space to sufficiently interpret unresponsive flows. It should be noted that FRED is memoryless, so that unresponsive flows will be considered as responsive once their packets are cleared from the buffer, limiting its stability.

Stochastic Fair BLUE (SFB) is an enhancement of the BLUE algorithm in terms of fairness [11, 55, 85]. It is a per-flow information algorithm that uses multi-level L hash table with N number of bins in each table. Each bin is assigned to a certain flow and counts how many times that flow has been hashed; associated with each bin is a dropping or marking probability P_m. For each L table there is an independent hash function which assigns a bin for each flow based on the flow ID (which contains its source address, destination address, source port, destination port, protocol). Each packet is hashed on its arrival, that increases its occupancy bin and in turn the P_m associated with that bin, and vice versa. A flow with a high transmission rate would increase its P_m value from 0 to 1 in all of its bins in the L tables. Thus, this flow is considered as non-responsive and its rate should be penalized, as shown in Figure 2.4.

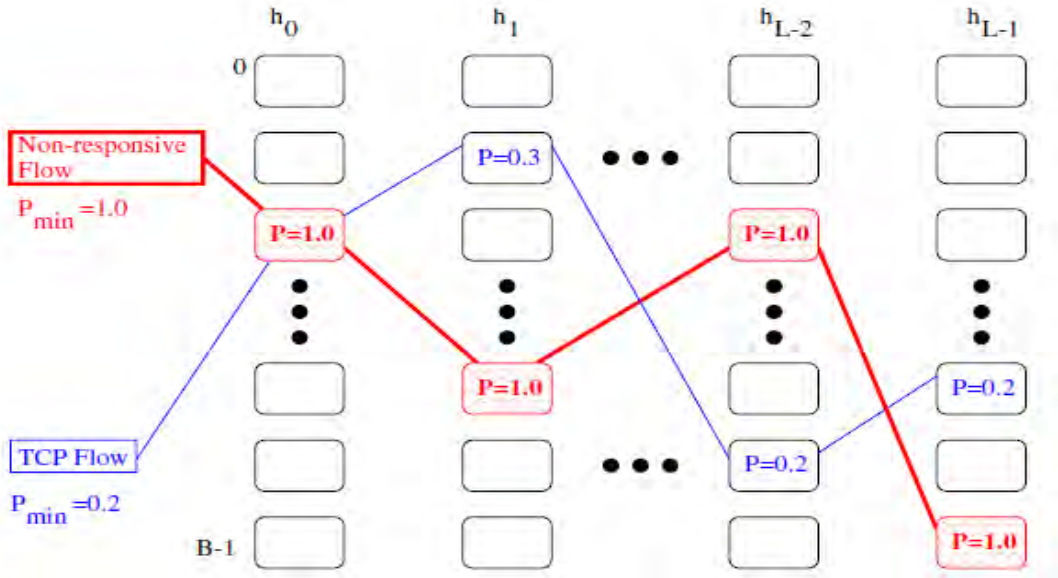

Figure 2.4. Multi-Level hash table in SFB [63]

The novelty of the SFB scheme is to protect responsive flows by penalizing non-responsive flows and limiting their rate of a fair share; it differentiates between responsive and non-responsive flows by using bloom filters with a multi-level hash function. The main disadvantage of SFB is misclassification. It has been approved by [16] that responsive flows may occupy some bins associated with non-responsive ones, which penalizes responsive flows as well. Thus, the concept of fairness and stability will be punished in the network. Nonetheless, since the SFB congestion indicator mechanism is independent of queue occupancy, some non-responsive flows will be penalized even when an empty buffer space is available.

BLACK: The main idea behind BLACK (from BLACK-listing unresponsive flows) is to control high bandwidth unresponsive flows with different types of flow to achieve fairness [91]. BLACK uses buffer occupancy fraction and bandwidth share as a congestion indicator at the link. Based on FIFO queuing, if the AQM allocates the buffer among all the flows equally, fairness can be achieved at the link bandwidth.

BLACK uses limited state information and a sampling technique to estimate the buffer occupancy fraction of only the large queue. To differentiate between flows, BLACK uses cache memory (High Bandwidth Flow or HBF cache memory) to capture the flow ID for all the flows that occupy the queue. As each packet arrives at the queue, BLACK records this packet in the HBF cache memory. Whenever the queue length exceeds a certain threshold, a packet is randomly picked from the queue and its ID compared with the ID of the incoming packet. If these two packets have the same ID then it is recorded in HBF cache memory and the *Hit* value is changed to 1. If the IDs are not same then it will be recorded in the memory and keep sampling. After *m* samplings, the *Hit Fraction* for the flow, which is responsible for estimating the buffer occupancy of that flow, can be calculated by dividing the *Hit* value by *m*. The flow with the higher *Hit Fraction* is considered as a high bandwidth flow and will be dropped according to the probability function, which can be calculated as:

$$\widehat{p_i} = \frac{\overline{H_i} - \frac{1}{N_{act}}}{(\frac{1}{N_{act}})} \tag{2.3}$$

where $\overline{H_i} = \frac{Hit_i + H_i m}{\acute{m} - m}$

$H_i: Hit\ Fraction\ for\ i\ flow = \frac{Hit_i}{m}$

$Hit_i: The\ number\ of\ Hits\ at\ the\ sampling\ time$

$\acute{m}: The\ number\ of\ samples\ taken\ so\ far$

$N_{act}: The\ estimate\ of\ the\ number\ of\ active\ flows$

$\widehat{p_i}: Dropping\ probability\ for\ flow\ i$

This dropping probability will be scaled according to RED congestion avoidance as:

$$p_{final,i} = \hat{p}_i \times \frac{\bar{q} - min_{th}}{max_{th} - min_{th}} \quad (2.4)$$

The flows that have no record in HBF cache memory are controlled totally by RED. When high bandwidth flows remain in HBF cache memory they are penalized according to how much greater than their fair share they occupy.

BLACK has been proved to perform fairly not only in high bandwidth unresponsive flow scenarios but also in scenarios where different TCP types and RTTs are competing for the link bandwidth [92]. The main drawback in BLACK is its inaccuracy in estimating the number of flows in most cases, such as when the queue size is not large compared with high bandwidth and delay applications, or when traffic transmission rates of flow are very different. In addition, as BLACK acts like RED in some cases as mentioned before, in that case it has the same drawbacks, such as each packet arrival checking and the probability function.

Table 2.3 compares the heuristic AQM schemes considered as RED enhancements with improved fairness.

Table 2.3

RED variants for improving fairness

AQM Scheme	Congestion Indicator	Special Characteristics
Fair RED (FRED)	Average queue length	- Penalizes unresponsive flows to protect responsive flows - Unfair in many cases, although fairer than RED - Needs large buffer space to sufficiently decipher unresponsiveness - Memoryless in that the flow is immediately reclassified as responsive once all its packets are cleared from the buffer
Balanced RED	Average queue length	- Attempts to protect adaptive flows from non-adaptive flows using minimal flow state information. - BRED was augmented with a virtual buffer to provide fairer bandwidth allocation - Drops packets on the virtual buffer

Table 2.3 Continued.

AQM Scheme	Congestion Indicator	Special Characteristics
Short-lived Flow Friendly RED (SHRED)	Average queue length	- Attempts to improve fairness for short-lived TCP flows - Flows with smaller congestion windows experience lower dropping rate - SHRED needs to know the current congestion window for each flow
CHOKe	Average (EWMA) queue length	- Tries to achieve fairness by using max-min fairness without any state of information - ECHOKe is an enhancement of CHOKe which replaces the RED mechanism with the REM mechanism
GREEN	Rate-based	- Needs to determine maximum segment size, RTT, and the number of active flows to work - Estimates maximum segment size by looking at the packet size - Estimates number of flows according to the time interval - If interval too long, the number of flows will be high and link utilization will drop - If interval too short, the number of flows will be underestimated and the allowed throughput will be higher than it should be
Stochastic Fair BLUE (SFB)	Rate-based	- Flow misclassification problems - The hash functions are changed and the bins reset at random times creating virtual bins across time - Unresponsive flows will behave badly with every reconfiguration
BLACK	Rate-based	- Estimating *Nact* is far from the reality - Needs to be enhanced in terms of fairness

2.3 Deterministic Optimization Approach

Optimization approach is one of the approaches used in AQM design, alongside the heuristic and control-theoretic approaches. Kelly et al. [52], [70] developed the optimization approach in 1998; it was extended by Low et al. [43] in 1999, the main difference between them being that the former optimizes the source rate and the latter optimizes the congestion measure. The optimization method can be divided into deterministic, stochastic and neural networks, all of them based on the optimization problem and analysis. Deterministic optimization attempts to push the network to the globally optimal performance point by using an AQM algorithm in conjunction with an end-point algorithm. To better understand the AQM schemes using this approach, the optimization problem itself should be understood. The non-linear programing problem considers perfect fluid flows as an optimization problem [93]. This problem can be formulated as [43], [52], [69], [70]:

Assume that a random network contains a set of links $\mathcal{L} = \{1, \dots l, \dots L\}$ each with its own capacity c_l where $l \in \mathcal{L}$. With network links there is a set of sources $\mathcal{S} = \{1, \dots s, \dots S\}$ and each source has its own transmission rate $x_s(t)$ where t is time in packet per second, so that the vector of all source rates will be $x(t) = [x_1(t), \dots, x_s(t)]^T$. For each individual source there is one link or set of links $\mathcal{L}_s \subseteq \mathcal{L}$ so that a source that uses link l can be expressed by $\mathcal{S}_l = \{s \in \mathcal{S} | l \in \mathcal{L}_s\}$. Each link has its own congestion measure scalar called "price" with the unit of bandwidth; price can be denoted by $p_l(t)$, and its vector will be $p = \{p_1(t), \dots, p_L(t)\}^T$. $\mathcal{R}$ is the routing matrix from $L \times S$; it can give a more suitable description for these source-link interdependencies as:

$$\mathcal{R}_{ls} = \begin{cases} 1 & if\ l \in \mathcal{L}_s\ or\ s \in \mathcal{S}_l \\ 0 & otherwise \end{cases} \tag{2.5}$$

However, the aggregation of all source rates at the link l is $y_l(t) = \sum_l \mathcal{R}_{ls} x_s(t) = \mathcal{R}x(t)$, so that the end-to-end congestion measure for the source will be $q_s(t) = \sum_l \mathcal{R}_{ls} p_l(t) = \mathcal{R}^T p(t)$.

When the rate $x_s(t)$ has been transmitted by the source s, it consumes some of the link utilization, donated by $\mathcal{U}_s(x_s)$. Assuming the utility function is increasing as the number of sources increases. Thus, the aggregation of the utility function for all the sources will be $\sum_s \mathcal{U}_s(x_s)$.

Therefore, the problem will be formulated as:

$$maximize \sum_s \mathcal{U}_s(x_s) \tag{2.6}$$

$$subject\ to \sum_{s} \mathcal{R}_{ls} x_s(t) \leq c_l \quad l = 1, \ldots, L \qquad (2.7)$$

The optimization problem can be solved by using the vector of source rates (x), that maximize the utility function ($\sum_s \mathcal{U}_s(x_s)$) constrained by the condition that the link capacity c_l will not be exceeded by the aggregate source rate at every link. However, there are two issues regarding this assumption. First, the utility function might differ among sources and will not be known by the network [52]. Secondly, solving the optimization problem requires potential coordination among all the sources, which is a complex and impractical situation because the rates are conditioned by constraint, although they are independent in the utility function [43]. In order to overcome these issues, a distributed solution has been suggested [68]. Each source adjusts its own rate and end-to-end congestion measure without any information from other sources or links in the network, and each link controls its own price independently.

From the optimization problem formulation, it can be noted that price can be measured using the source rate not the queue length. Thus, each AQM design based on this formulation will be rate-based not queue-based [94]. Three AQM schemes have been designed using the deterministic optimization approach, AVQ, SVB and REM, discussed in depth below.

i. Adaptive Virtual Queue (AVQ)

AVQ [8] is a rate-based AQM that maintains the arrival rate at a targeted utilization [12, 13, 27, 32, 45, 75, 90]. It was designed by Kunniyur and Srikant in 2004 [8], based on Kelly et al.'s [70] optimization approach. The probability function of AVQ is derived from M/M/1/B loss probability (see [96]) which is $\mathcal{P}_l(c_l, \mathcal{Y}_l) = \frac{(1-\rho)\rho^B}{1-\rho^{B+1}}$ where

$\rho = \frac{y_l}{c_l}$ is link utilization, and B is the buffer limit. In this loss probability, B, link capacity (c_l), and arrival rate ($\mathcal{Y}_l$) are scaled by K factor and take the range $K \to \infty$, so the probability will be $\rho_l(c_l, \mathcal{Y}_l) = \frac{(\mathcal{Y}_l - c_l)^+}{\mathcal{Y}_l}$ where $[z]^+ = \max(0, z)$.

AVQ adapts a virtual queue with link capacity less than the physical link capacity connected to the real queue [97]. The packet is marked when the arrival rate at the queue exceeds the virtual capacity [96]. The actual number of flows should be known by the queue in order to compute a virtual capacity that will satisfy the probability function. This computation should occur when the network load is changing because the number of flows varies over time. However, the link capacity of the virtual queue is calculated by a differential equation to make the computation independent for each number of links, as:

$$\tilde{C}(t) = \alpha\big(\gamma C - \mathcal{Y}(t)\big) \tag{2.8}$$

where

$\tilde{C}$: *The link capacity in the virtual queue.*

C: *The link capacity in the real queue.*

$\mathcal{Y}(t)$: *The arrival rate to the system*

α and γ are AVQ parameters. γ is the desired link utilization and α is a smoothing [8] or step-size parameter [96]. α is considered as a key design issue for AVQ because it determines the adaption speed of the virtual link capacity, while γ represents how robust the system is in the presence of short-lived flows [95]. As stated by [8], the stability of the system can be determined by both α and γ. It is readily seen from

Equation (2.8) that AVQ tries to match the arrival rate with the link capacity of the virtual queue to achieve the desired level of utilization [17].

The size of the real and virtual queues is the same, and both queues will receive same arrival rate, although the virtual queue will build up and overflow faster than the real queue due to its parameters. Each time the virtual queue experiences overflow, the same packet is marked/dropped in the real queue [15]. This concept is deterministic and is the opposite of the probabilistic concept used by RED [83]. Eq. (2.8) also indicates that when the virtual link capacity is greater than real link utilization, the marking probability will be aggressive, and vice versa [8].

The dropping probability for AQM [17], will be:

$$p(t) = \left[1 - \frac{\bar{C}(t)}{y(t)}\right]^{+} \tag{2.9}$$

By making the value of γ equal 1 and applying it in Eq. (2.8) with initial settings $\tilde{C}(0) = C$ and $q(0) = 0$ ($q(t)$ the queue length at time [17], the differential equation is:

$$\tilde{C}(t) = C - \alpha q(t) \tag{2.10}$$

This indicates that AVQ matches the virtual capacity against the queue length, so when the queue length increases, the virtual capacity also increases.

AVQ has shown a significant performance especially in high link utilization, low packet loss, and low delay [15]. The virtualization technique of AVQ improves performance in terms of robustness, responsiveness and stability. In fact, AVQ has been performed in stability analysis by the designers [8]. The result from this analysis

was some recommended rules to control AVQ parameters (α and γ) according to the number of flows for efficient performance. AVQ can be considered as a fair scheme as its control function maximizes the aggregation of source utilities in the network in the absence of feedback delays. The only disadvantage of AVQ is that it cannot differentiate unresponsive flows from responsive one because it was not designed to handle responsive flows [12, 13].

ii. Stabilized Virtual Buffer (SVB)

SVB [39] is a hybrid AQM that uses queue length and arrival rate as its congestion indicator, approximating both to individual desired values. It is like AVQ in the use of a virtualization technique in order to calculate the dropping/marking probability function, but SVB differs with AVQ in the following two major ways:

a. The virtual queue capacity in AVQ (*C*) changes dynamically according to the arrival rate, and the virtual buffer limit is fixed at the same level as the real buffer limit (*B*). In SVB, the virtual capacity is the same as the real capacity and is a fixed value, and the virtual buffer limit is adjustable according to the arrival rate.

b. In AVQ the packets in the real queue are deterministically marked/dropped when the virtual queue overflows. In SVB the packets in the real queue are marked with a probability according to the current virtual capacity and virtual buffer limit.

The virtual buffer limit in SVB can be determined as:

$$b_v(t) = \gamma\big(C - Y(t)\big) \tag{2.11}$$

SVB has been designed to enhance the stability, it achieved better performance than AVQ in terms of stability, and has shown some result in terms of robustness and responsiveness. However, SVB has given poor result in terms of fairness comparing with AVQ, and AVQ achieved better stability in wireless networks as well.

iii. Random Early Marking (REM)

REM was designed by Athuraliay and Low [35] to attain high link utilization with negligible loss and delay, in a stable and simple manner. The main difference between REM and RED is the congestion measure and the probability function for dropping/marking. The key idea behind REM is achieving its targeted queue length for low delay and targeted rate for high utilization independently of network congestion [98], by decoupling the congestion measure from the performance measure [15], [20], [35], [81], [97], and reaching the global optimal performance point [81].

The congestion measure used by REM, known as 'price', is the weighted sum of the mismatch between the arrival rate and queue length and the targeted ones (the difference between arrival rate and link capacity and the difference between the queue length and the targeted length). The price is updatable individually for each link; when the aggregation of this weighted mismatch is positive, the price will increase, and otherwise decrease. Whenever the arrival rate exceeds the link capacity or the queue length is greater than the target, the weighted sum for the mismatch will be positive, otherwise negative. The incrementing of the price pushes up the marking probability, sending a strong signal to the sources to reduce their rates. Reducing the arrival rates reduces the price and hence the marking probability, until eventually the weighted sum

for the mismatches is zero. At this point REM achieves the highest link utilization and minimum delay and loss. The REM price can be updated as in the following equation [29, 78]:

$$\mu_l(t+1) = [\mu_l + \gamma(\alpha_l(q_l(t) - q_l^*) + x_l(t) - c_l)]^+ \qquad (2.12)$$

where

μ_l: $The\ price\ of\ link\ l$

q_l: $The\ actual\ queue\ length$

q_l^*: $The\ target\ queue\ length$

α_l: $Stability\ constant$

$[z]^+ = max(z, 0)$

The exponential packet marking probability for the *l* link is:

$$p_l(t) = 1 - \emptyset^{-\mu_l(t)} \qquad (2.13)$$

where

p_l: $Marking\ probability\ for\ the\ link\ l$

$\emptyset$: $constant\ \emptyset > 1$

Thus, the end-to-end marking probability can be expressed as [35]:

$$1 - \prod_{i=1}^{L}\big(1 - p_l(t)\big) = 1 - \emptyset^{-\sum_l \mu_l(t)} \qquad (2.14)$$

This means that the end-to-end marking probability will increase as the individual prices increase.

REM was designed to perform optimally in a steady-state situation, but is less efficient in a transit situation [23, 29]. However, when the designers ran the stability test on REM they found that prior knowledge about the network parameters, such as the number of flows and RTTs, could guarantee sufficient stability for responsiveness, although these parameters are frequently changed in real networks. REM performed badly in the wireless tests, and experiment that has been done by the REM designers, and they claimed that the reason behind this poor performance was because TCP sources cannot differentiate between the overflow loss or wireless effects loss, and therefore reduces the transmission rate in both events. Table 2.4 summarises the comparison between optimization approach schemes.

Table 2.4

Comparison of Optimization Approach Schemes

AQM scheme	Congestion Indicator	Control Function	Characteristics
AVQ	Rate-based Price equation: $\tilde{C}(t) = \alpha(\gamma C - y(t))$	$p(t) = \left[1 - \frac{\tilde{C}(t)}{y(t)}\right]^+$	- high link utilization, low packet loss, and low delay - virtualization technique gives AVQ better performance in terms of robustness, responsiveness and stability - AVQ parameters (α and γ) need to be controlled - AVQ cannot differentiate unresponsive flows from responsive
SVB	Hybrid (queue length & arrival rate) Price equation: $b_v(t) = \gamma(C - y(t))$	Same as AVQ	- Enhances stability - Poor result in terms of fairness compared with AVQ in wireless networks
REM	Hybrid (arrival rate & queue length Price equation: $\mu_l(t+1) = [\mu_l + \gamma(q_l(t+1) - (1-\alpha_l)q_l(t) - \alpha_l q_{ref})]^+$	$p_l(t) = 1 - \emptyset^{-\mu_l(t)}$ End-to-end marking probability $1 - \prod_{i=1}^{L}(1 - p_l(t)) = 1 - \emptyset^{-\sum_l \mu_l(t)}$	- Designed to perform optimally in steady-state situation - Prior knowledge about the network parameters guarantees stability, but these parameters are frequently changed in real networks - Poor performance in wireless networks

2.4 Controlling Queue Delay (CoDel)

CoDel is the latest AQM, proposed by Nichols and Jacobson in 2012 [31]. It was designed to solve the full buffer problem, "bufferbloat", in networks by limiting the packet queue delay in the network links (routers). CoDel aims to enhance overall performance of the network by reducing the delay and packet loss with high link utilization and throughput. According to [31], it has significant characteristics that make it better than other AQMs, such as

- Parameterless: no pre-configured parameters required.

- Treating good queue (the queue that drains as fast as possible) and bad queue (the queue that fills up at same as transmission rate) differently.

- Queue delay is controlled regardless of RTT delay and traffic load.

- Maintaining dynamically changing send rate without any effect on link utilization.

- Simple to implement in real router.

CoDel can be considered as a delay-based AQM because it uses packet-sojourn time instead of arrival rate or queue length in its congestion indicator. Packet-sojourn time is the time that the packet spends in the queue, which can be found by adding a timestamp to each packet arriving at the queue that contains arrival time for that packet. When the packet is about to leave the queue, the packet-sojourn time is simply calculated by subtracting the leaving time from that recorded in the timestamp (arrival time) for each packet independently. If the sojourn time is longer than a pre-defined

target, the algorithm sets a timer for dropping the packet on dequeueing (leaving the queue). This drop will occur only when the sojourn time is bigger than the target and the packets in the queue are fewer than one Maximum Transmission Unit's (MTU's) bytes. The time indicating the next dropping event is updated periodically according to the equation below:

$$Next\_drop\_time \mathrel{+}= {}^{interval}/_{\sqrt{count}} \tag{2.15}$$

The count represents the total number of dropped packets since the first drop event. Interval is the minimum value of the sliding window that entered the queue; CoDel has to identify this by time, because it is varies with time, and update it frequently. The dropping action is stopped when the sojourn time goes below the target value.

CoDel has two important parameters, target and interval, which must be configured carefully for better performance. However, these parameters are given fixed values, chosen based on many simulations and experimental results, [99] as follows:

- *Target*: constant 5ms (acceptable queue latency)

- *Interval*: constant 100ms (in worst case of RTTs)

CoDel has shown better results in many comparisons with the previous AQM schemes. Raghuvanshi, Annappa and Tahiliani [100] compared CoDel with RED and Adaptive RED (ARED), and concluded that CoDel is independent of queue size, rate measurements, drop rate and RTT delays, and they showed that CoDel has better performance in link utilization, queue length, and drop rate, but the authors suggested that CoDel needs more optimization and improvement to increase its robustness.

According to [101], CoDel performs better than droptail and RED algorithms in terms of queue delay, link utilization and packet drop. However, it has been concluded that CoDel has a higher packet loss than RED when the number of flows increases. This issue is not acceptable in terms of network stability, and CoDel needs to be improved to control stability as the number of flows increases. In terms of fairness, CoDel is considered better than some of the RED variants, but it needs to be enhanced by combining the CoDel algorithm with a scheduling algorithm (such as Fair Queue) [102].

Table 2.4 compares the algorithms discussed above, with the most important characteristics that affect fairness and stability in AQM algorithms. From the table, we can see that congestion indicator type is important for fairness because the more accurate the indicator the greater degrees of fairness between the different flows. It can be seen that, the rate-based congestion indicator gives the best result than the rest. Moreover, the hybrid congestion indicator gives a better result than other indicator in terms of fairness. In addition, algorithms with a specific mechanism to differentiate between flow types or RTT flows achieve better fairness than the others.

Table 2.4 also shows that algorithms with manual parameter configuration directly affect stability: the more accurate the configuration, the greater the stability in the network. Unresponsive flows should be penalized to increase stability in the network.

Table 2.4

AQM Schemes Comparison Table

AQM	Congestion Indicator	Parameter Cpnfigurations	Constants Values	Differentiate Between Flows	Penalize Nonresponsive Flows	Maintain Different RTT Flows	Stability	Fairness	Note
RED	Queue-based	√	-	-	-	√	-	-	Fairer than droptail
FRED	Average Queu Length	√	-	√	√	√	-	-	Fairer than RED
SFB	Rate-based	√	-	√	√	√	-	√	Differentiate between flows by hash table
BLACK	Rate-based	√	-	-	-	√	-	√	Inaccurate to estimate the number of flows
AVQ	Rate-based	√	-	-	√	√	√	√	Cannot differentiate nonresponsive flows from responsive
SVB	Hybrid (queue length & arrival rate)	√	-	-	√	√	√	-	Achived better stability but less fairness than AVQ
REM	Hybrid (queue length & arrival rate)	√	-	-	√	√	√	√	Poor performance in wirless network
CoDel	Queue Delay	-	√	-	√	√	-	√	Need to improve the stability and fairness

2.5 Summary

This chapter provides the detailed background on issues that are covered in this thesis. The background materials on AQM schemes mechanisms and designing approaches were reviewed and critically analyzed for content and significance. Furthermore, it highlighted the main parameters that have been used in the congestion indicator and control function mechanisms detailed characteristics of each parameter. By analyzing the previous AQM schemes, the Optimization Approach has been identified as the designing approach for the proposed scheme. Beside, queue delay and source rate have been chosen as congestion indicator parameters with exponential probability function as control function. In the next chapter, the research methodology for achieving the objectives of this research and evaluating the proposed scheme will be presented.

CHAPTER THREE
RESEARCH METHODOLOGY

This research aims to design a new hybrid Active Queue Management (AQM) algorithm named (HFAQM) for infrastructured wireless network, specifically it detects and avoid congestions with providing fairness and stability in wireless network environment. However, designing and evaluating the performance of the proposed algorithm are important tasks to accomplish as mentioned in chapter one. In order to achieve this research objective, it is extremely important to apply an appropriate methodology to follow. For this purpose, this research adapts the Design Research Methodology (DRM) and produces its main steps according to the research objectives.

3.1 Research Approach

This research aims to design a new congestion indicator and control function with feedback signal mechanisms for AQM, named HFAQM, by enhancing existing AQMs to provide more stability in WLAN networks. This demands suitable planning between understanding the existing algorithms and designing a new one leading to a better solution [103]. These requirements are matched with the definition of Design Research Methodology as stated by Blessing [104]. It has been stated that design research should be scientific in providing valid results, both theoretically and in practice.

DRM contains a set of approaches and guidelines used as a framework for designing a research project; “it helps making design research more rigorous, effective and efficient, and its outcomes academically and practically more worthwhile”. In addition to these attractive objectives, DRM has others, listed below:

i. Provide a variant approaches that can fit different researches areas.

ii. Help researchers to conduct design research in a worthwhile and realistic manner.

iii. Combine a variant of the research types to gain more reflection on the applied approach.

iv. Provide pointers that are useful in differentiating aspects involved in the design.

DRM has four steps which are Research Clarification (RC), Descriptive Study-1 (DS-1), Prescriptive Study (PS) and Descriptive Study-2 (DS-2). DRM has been conducted in this research as can be shown in Figure 3.1 which describe the DRM from the perspective of this research with the methods been used in each step besides the expected outcomes from the steps.

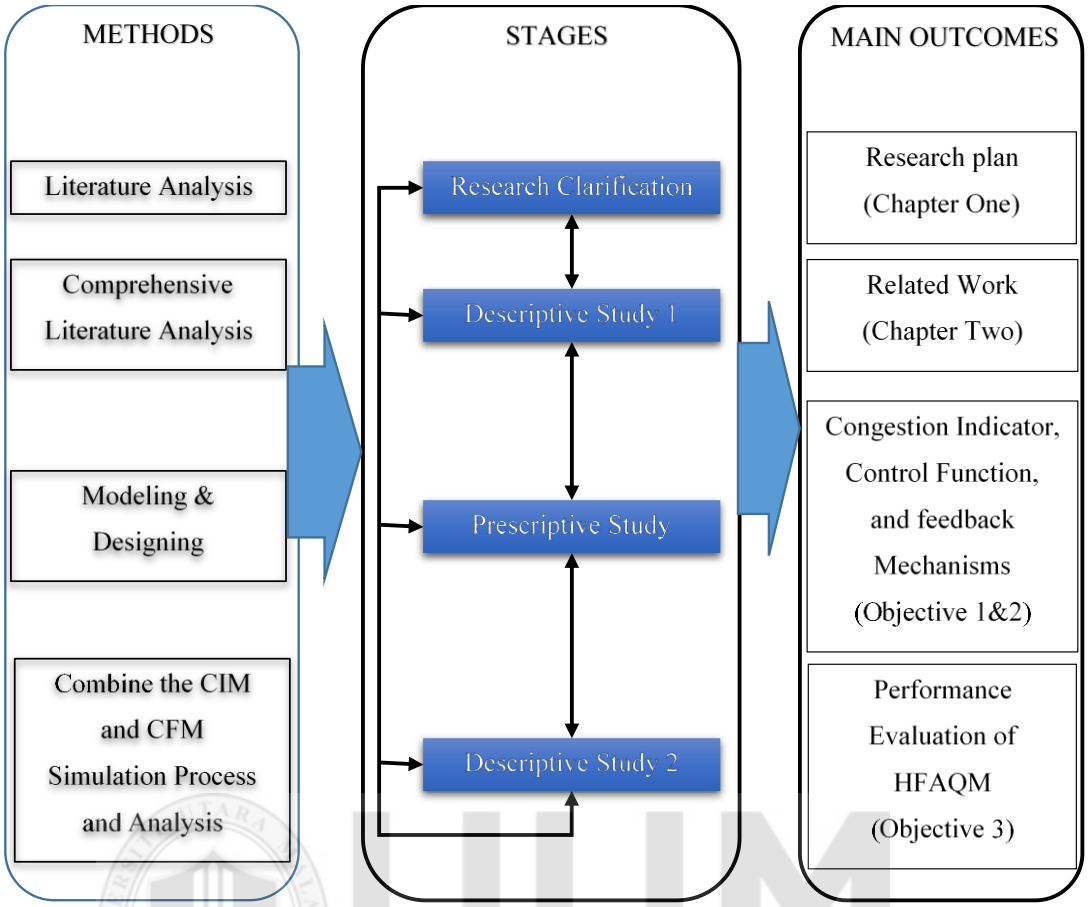

Figure 3.1. Research Approach

3.2 Research Clarification (RC)

RC contains six iterative steps that help the researcher to attain a deep understanding of the research area, recognize the challenges of the research, and design the overall research plan, as shown in Figure 3.2.

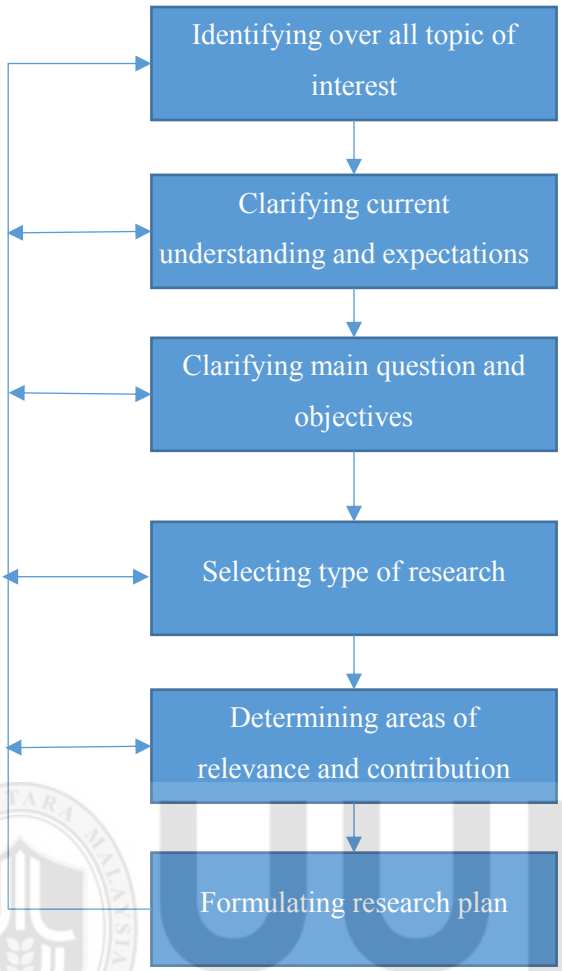

Figure 3.2. Research Clarification Stage
Source: Blessing & Chakrabarti [104]

In general, the output of this stage was presented in Chapter one, with definition of the specific outcomes as follows:

i. Research problems and motivation.

ii. Research objectives and questions.

iii. Related aspects to be considered.

iv. Overall research approaches.

3.3 Descriptive Study-1 (DS-1)

DS-1 is the second stage of DRM that focuses on providing a deeper understanding of the research area and the outcomes from the RC stage with a sufficient understanding of the current situation and development of the overall research plan. There are five steps in this stage as presented in Figure 3.3, reviewing the literature comprehensively with emphasis on the deliverables of empirical studies to identify the design approach, choose the congestion indicator parameters, and identify a suitable control function.

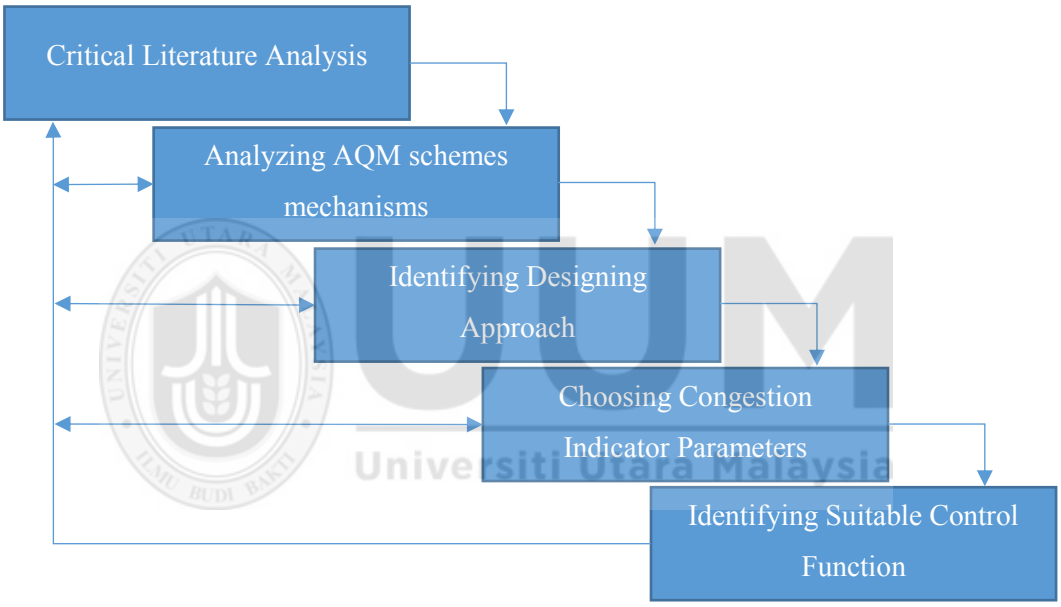

Figure 3.3. Descriptive Study 1 stage
Source: Blessing & Chakrabarti [104]

After analyzing the literature critically and examining related AQM mechanisms, different approaches to designing a AQM scheme were considered. The optimization approach was chosen to design HFAQM, combining two mechanisms: congestion indicator and control function. The congestion indicator mechanism will use hybridization of incoming transmission rates and queue delay as parameters to indicate

congestion. From the diverse types of control function, HFAQM selected the exponential function, described in detail in Chapter 4.

3.4 Prescriptive Study (PS)

The PS is the main stage in DRM, involving the design of the two proposed mechanisms (congestion indicator and control function with feedback signal) for HFAQM. This stage has six steps, from design of the mechanisms through computer programming to verification and validation of the mechanisms, as shown in Figure 3.4. The main outcomes from this stage are:

i. Achievement of Objectives 1 and 2 individually.

ii. Design and implementation of the proposed mechanisms.

iii. Validation and verification of the programming block.

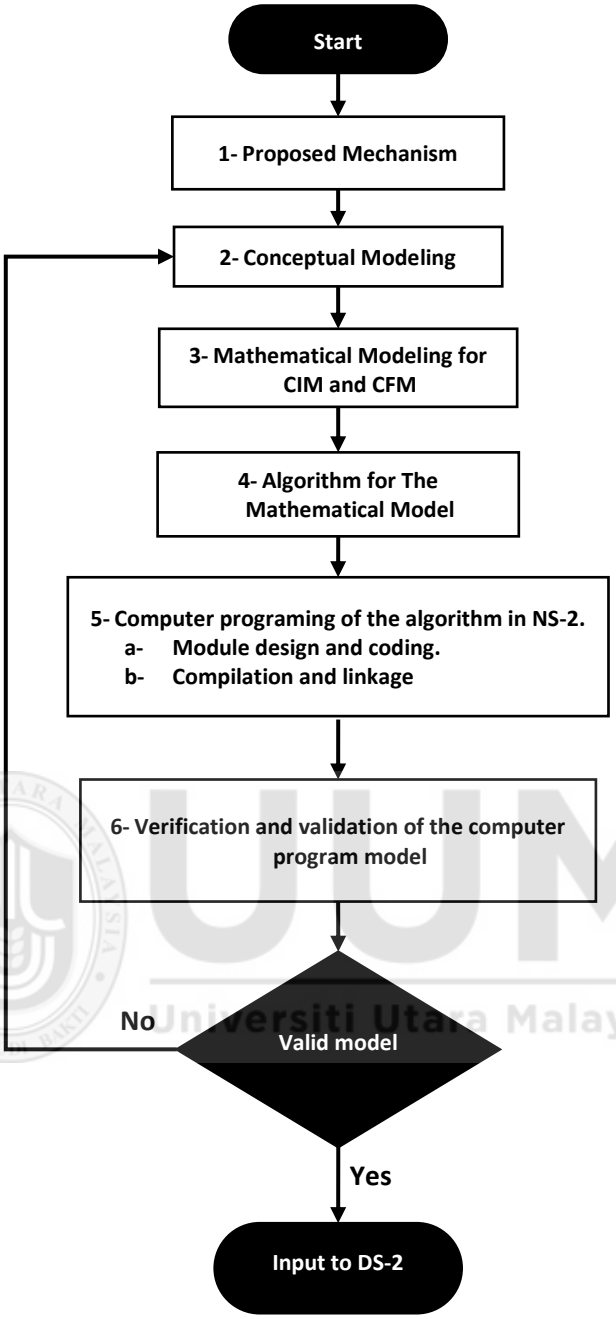

Figure 3.4. *Main steps in the PS stage*

3.4.1 HFAQM Conceptual Model

Based on the critical review of the literature and the research main contribution which is proposing and implementing AQM algorithm in wireless networks, congestion avoidance was determined as the key factor of this research. In practice, the main

objectives are improving the AQM components, congestion indicator and control function with feedback signal mechanisms, in order to provide fairness among different types of flow with stability in wireless networks. Thus, the conceptual model of HFAQM was designed to combine these mechanisms and provide the desired performance that needed to be achieved. In order to design each mechanism (objective), it is discussed and a design flowchart for each mechanism presented, as follow:

A. Congestion Indicator Mechanism (CIM):

The Congestion Indicator Mechanism (CIM) calculates the congestion level in the queue and tries to avoid congestion. It uses a mathematical formula to calculate the congestion level depending on various network parameters; this mathematical equation has different names in different AQM schemes, for example in RED it is the EWMA equation, and in AVQ called Price. Each AQM scheme also uses different network parameters in its equation, and the network parameter could be incoming rate, queue size, average queue length, queue delay, load, packet loss, or any combination of these.

From this research point of view, “price” refers to the mathematical equation to calculate the congestion level, according to the incoming rate and queue delay. The overall process for this congestion indicator equation can be understood through the flowchart illustrated in Figure 3.5. At the start of the process the AQM scheme initializes the values for both Target Queue Delay ($d_l^*(t)$) and incoming sources rate. These two values are compared with delay and link capacity; increasing fairness among different type of flow is the main expected achievement of CIM.

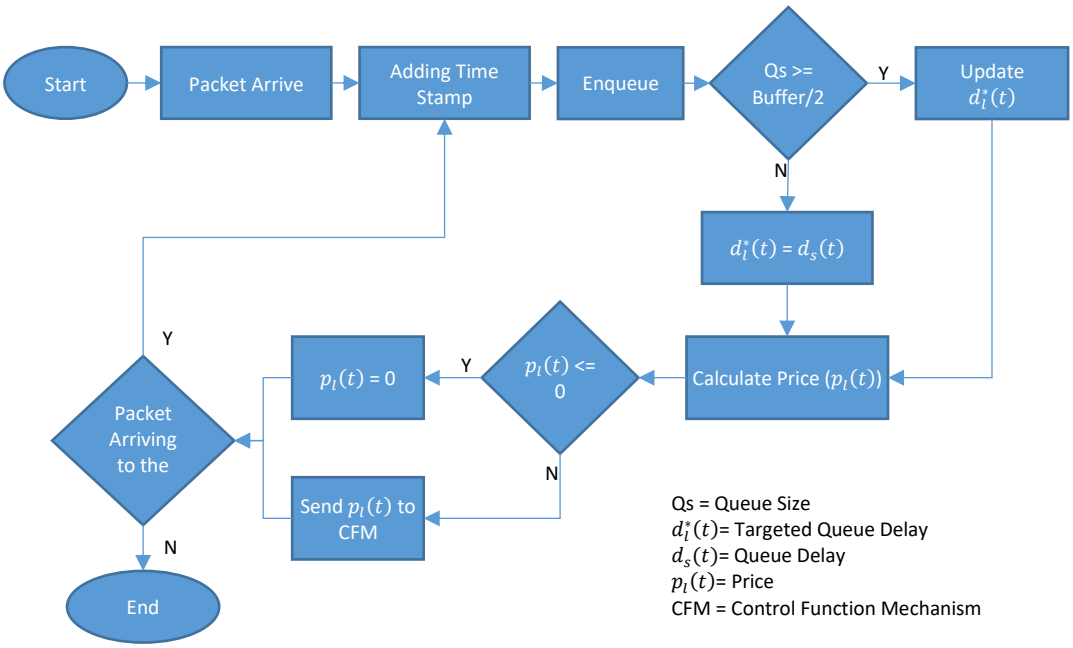

Figure 3.5. Congestion Indicator Mechanism Flowchart

B. Control Function Mechanism (CFM)

The Control Function Mechanism (CFM) is the probability function for marking or dropping a packet, which drives the congestion level as given by CIM. There is an algorithm associated with the control function that marks or drops the packet. This algorithm determines which packet have to be dropped or marked and from which port.

According to the research objective, the control function algorithm will increase stability in the WLAN environment. It uses hybridization between marking and dropping to control the aggressiveness of the probability function. In order to achieve stability, CFM penalizes non-responsive flows that have no ECN capability by dropping their packets according to their share of link capacity and the queue delay time and marking the responsive flows. When the queue length reaches the size of the buffer, both responsive and non-responsive flows are penalized to increase the aggressiveness and overcome the overflow situation as can be seen in Figure 3.6. The

outcomes from CFM are: increasing stability, low dropping of responsive flows, and high link utilization.

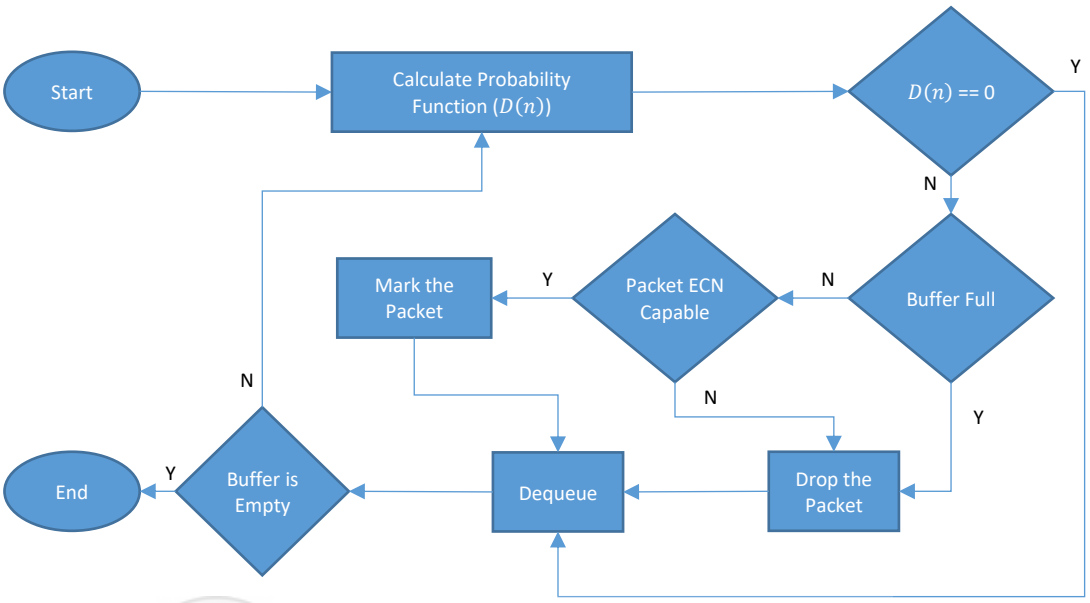

Figure 3.6. Control Function Flowchart

The conceptual model of HFAQM algorithm comprises of the two proposed mechanisms. The integration of congestion indicator mechanism and control function mechanism is forming the HFAQM scheme, whereas the congestion indicator mechanism hybridizes the queue delay with arrival rate. The overall conceptual model is shown in Figure 3.8.

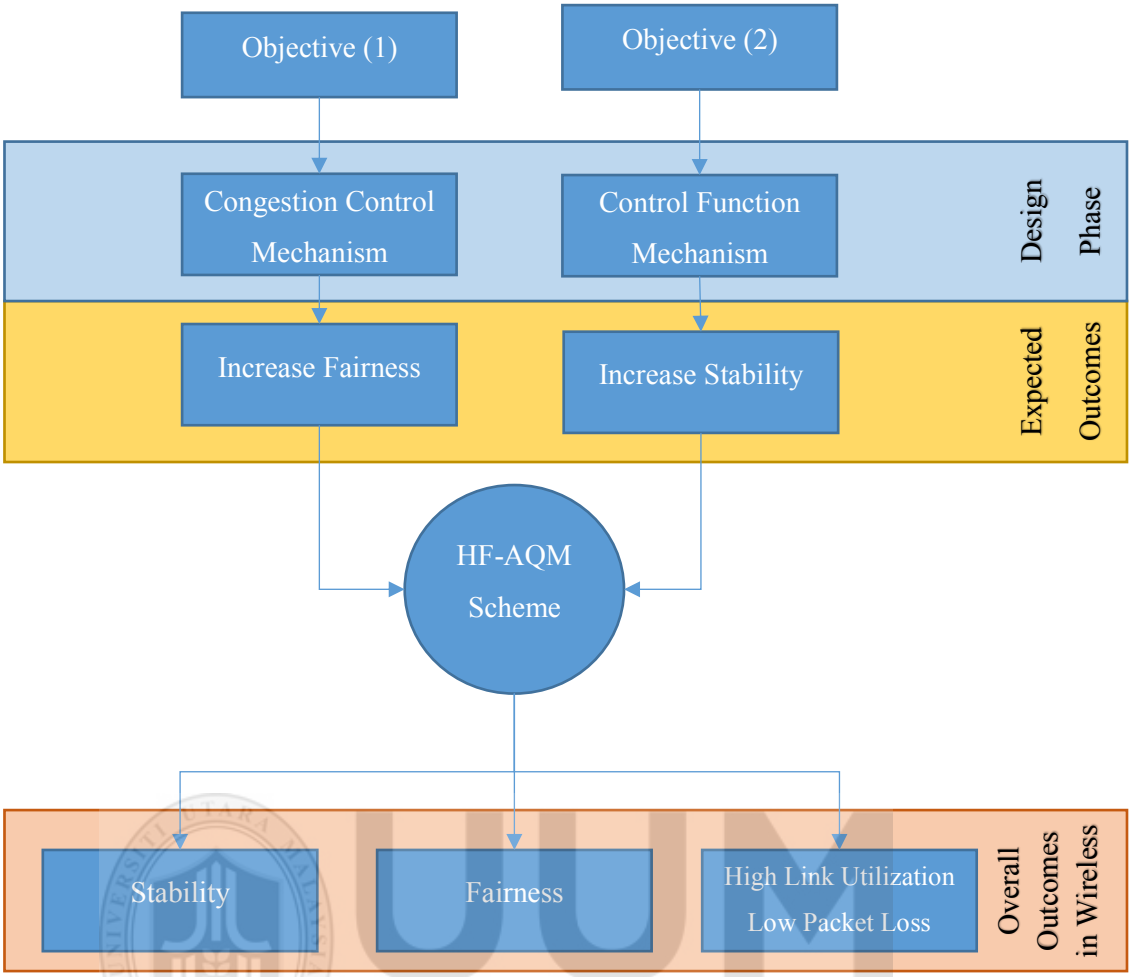

Figure 3.8. Conceptual Model for the Proposed HFAQM

3.4.2 Verification and Validation

Model verification evaluates the integrity of the transformed model that was illustrated through flowchart or pseudo code to an executable computer program [105]. The simulator used in this research was built based on C++. Therefore, in this research, the structure of the proposed mechanisms will be implemented using C++ as a programming language. In addition, all mechanisms must be verified to assure that code was written correctly without errors or bugs [106]. The main reason for conducting verification is to ensure that the proposed CIM and CFM are implemented properly in the NS-2 simulation environment, and is programmed correctly in C++ programing language. CIM and CFM will be verified separately in the next chapter.

Balci in [105] defined validation as the affirmation of the implemented model, policy or a mechanism that acts accurately compared to other validated models. The accurate ratio has to be acceptable and the behaviour has to be consistent with models and simulations. Hence, the validation perspective is about to build a right model. The right model means that the introduced mechanism performs the basics and required functions properly.

Validation techniques were discussed in [105], [107]. These methods are informal, formal, static and dynamic. Each main method has sub-techniques. The informal method depends on the human factor; neither rigorous mathematical rules nor guidelines exist in this method. Informal method is usually applied in robust approaches based on formal guidelines. Audit technique [108] is one example of informal method.

The formal method is based on mathematical proof. If the mathematical model of proof is correct, that means the model is valid. However, not all models are obtainable to prove mathematically their correctness. In addition, based on the current state of the art, many formal techniques are not applicable for complex and reasonable simulation models. Inductive and inductive assertions are examples of formal method techniques [108], [109].

The static method concerns about the truthfulness of model's aspect. This method does not need an actual implementation of the model. Rational implementation is sufficient to validate the model. The static mode has information about the flow structure of the model, the source code and data. This method can be implemented for automated tools.

The simulation compiler represents [108] and Cause-Effect Graphing are examples of this method [110], [111].

Dynamic method requires an implementation for the model. The validation is based on the behaviour of the implemented model. The dynamic method is implemented through three phases. The first phase is adding a source code into the body of the executable model. The inserted codes collect information about the implemented model. Then, the model is executed. Finally, the output of the executed model will be compared with other valid models. The verdict of the validation will be based on the behaviour of the output model compared to the valid models [105], [112]. Alpha testing [113] technique is an example of this method.

Since, this study focused on designing AQM in dynamic WLAN environment, the dynamic method will be implemented to validate the CIM and CFM. The proposed mechanisms output will be compared with the output of other validated mechanism (REM) using same simulation tool based on the same experiment setup and environment. The validation will be implemented by simulating different scenario, each scenario will be designed to measure each mechanism intended purpose. The result that are generated from simulating CIM and CFM will be presented graphically. Then, the behaviour of the proposed mechanisms will be compared with valid mechanism (REM) based on the graphical lines [114], [115]. The validation of CIM and CFM touches three techniques in dynamic approach, which are Sensitivity Analysis [116], [117], Graphical Comparison [118], [119] and Comparison Testing [120].

3.5 Descriptive Study-2 (DS-2)

The last stage of DRM is DS-2, which focuses on evaluating the proposed mechanisms and overall HFAQM algorithm. The performance evaluation is very important step and there are many methods that have been improved and developed to solve the evaluation step. Mathematical modeling, measurement, and simulation are the traditional approach that have been proven and used by many researchers in terms of the performance evaluation [121], [122].

3.5.1 Evaluation Methods

At this point, choosing the evaluation technique is very important and crucial in order to evaluate the performance of the proposed algorithm and mechanisms. This section reviews the three traditional evaluation approaches (mathematical modelling, measurement and simulation) that are widely used to evaluate the performance of new mechanisms and highlight their strengths and weaknesses.

3.5.1.1 Mathematical Modelling

Mathematical modelling (also called analytical modelling) is a set of formulated mathematical equations that describe the actual system behaviour and performance [123]. By translating the functional relationships of the system into operations, computer programming can be used to investigate the mathematical model. The graphical model is the main representation for output, with theorems to prove and evaluate the model results, in which the user can easily adjust the input variables and parameters to gain a better result. Although mathematical models can be used to represent simple systems, it becomes too hard to represent complex systems mathematically without reducing the system accuracy. Although loss of accuracy is

the main disadvantage, according to Jain [124] the approach has advantages: less time required, low cost to use, and easy to evaluate.

3.5.1.2 Measurement

Measurement (also called experimentation) approach is testing the system performance via a test-beds or implementation on real network devices. By using this approach, the user can implement and design the system in real environment and the system performance can be evaluated based on real data that is collected from real devices by using emulator or special network software [125]. Furthermore, the accuracy in this technique is very high, but still have difficulties in implementation in terms of expensive cost and constructing a real network. Regarding to this research, it is quite difficult to provide a configurable wireless devices and routers at lower cost.

3.5.1.3 Simulation

Simulation is the most widely used approach among the others. It can be defined as an approximate representation or abstraction of real devices, systems or environments [113, 114]. Network simulation is a computer-generated system that designs, executes and analyzes the output of the theoretical communication network system [128]. The simulation approach makes it possible to examine the algorithms' performance in reliable, scalable, controllable and heterogeneous environments [109, 110, 116]. For this reason, the simulation approach is used as the main evaluation technique for this research. Its advantages include low cost, moderate time consumption, easy implementation of complex topologies, and a wide range of output data that covers all the experimental events. Table 3.1 compares the three techniques.

Table 3.1

Comparison of Performance Evaluation Techniques

Criteria	Analytical modelling	Simulation	Measurement
Time required	Low	Medium	High
Accuracy	Low	Moderate	High
Tools	Analysis	Computer languages	Instrumentation
Trade-off	Easy	Moderate	Difficult
Cost	Small	Medium	High

Source: Raj Jain [124]

3.5.2 Simulation Tools

Simulation has been chosen as performance evaluation technique for this research, as mentioned before. It has been used successfully for implementing, analyzing and applying AQM algorithms in wireless network environments [8], [14], [35], [39]. Simulation software used in this research domain includes NS-2, OPNET and OMNET++. Each of these simulators has its own advantages and disadvantages, reviewed below in the light of the requirements of this research.

OMNeT++ is an open source simulator software which is built based on modules and a Graphical User Interface (GUI) that can simulate general purpose discrete events. Its generic and flexible architecture make OMNeT++ a general simulator, not specifically for networks, although it allows configuration of concurrent multiple network layers from Link to Application layers.

OMNeT++ modules communicate with each other through exchanging messages. They can be combined to form new modules with combined features. The construction of a new system model should be structured in NEtwork Description (NED) language and mapped with communicating modules hierarchically; it can be modified and

reconfigured using the code editor or GUI based on Eclipse and called "Integrated Development Environment (IDE)". C++ is used to program each module with a class library and kernel connected to the OMNeT++ kernel. The simulation results are exported as text-based files and saved as vector or scalar files which can easily be analyzed using Matlab or other tools.

OMNeT++ has attracted attention in the academic world because of its hierarchical module structure and powerful GUI. However, the difficulty of building a new module and linking it to the OMNeT++ kernel is the main disadvantage of this tool.

OPNET is commercial GUI software that combines three main functions: modelling, simulating and analysis. These three functions make it a very powerful tool for simulating different types of network and distributed systems. The analytical model built based on mathematical models gives OPNET Modeler speed and accuracy in finding a result and handling complicated system sensors. The Modeler comes with more than 400 predesigned protocols, algorithms, devices and environments which make it a very powerful program that can be used for urgent and immediate situations. The simulation results can be studied and analyzed directly using the analysis model already built into the OPNET Modeler.

OPNET has produced good results in industrial research needing a group of built-in models to simulate complex scenarios within a short time and with accurate results. However, it is not open-source software and is very expensive; building one's own algorithm is difficult, although some examples are given.

NS-2 is Network Simulator version 2, the second version of Network Simulator (NS) developed in 1989 by University California Berkeley (UC Berkeley) as part of the

Virtual Internet TestBed (VINT). NS-2 is a very popular software used for simulating advanced algorithms and protocols in network research. It is an open-source object-oriented discrete-event simulator that can simulate realistic network topologies and characteristics. It is written in C++ and object-oriented Tcl (OTcl) scripts that are interpreted. Network components such as devices, links and protocols; and network characteristics such as delays, are represented by various classes.

The network model in NS-2 is constructed by connecting several components, calls NS objects. These objects include nods, classifiers, links, queues and many more. They have the advantage of the hierarchical structure of C++ classes already built in, which makes maintaining or building one's own algorithm and protocol comparatively easy. The text-based output results from NS-2 can be studied and analyzed using AWK or Perl script languages.

NS-2 is the most popular simulator tool due to its research libraries and environment for analyzing and testing advanced network simulations. It is available for all researchers with a wide range of supporting resources that include a good manual, available mailing list, source codes and examples.

As mentioned above, choosing a simulation tool is a very critical issue. Regarding to this research requirements, the above-mentioned simulation tools are the most widely used in simulating and evaluating AQM algorithm performance. Many studies have compared them in order to identify the right tool for the research domain and requirements, based on time, result accuracy, supporting libraries, availability, and cost.

According to Koksal's comparative study [130], NS-2 and OMNeT++ are suitable

choices for network research. NS-2 is the more widely used for implementing and studying one's own AQM algorithms. On other hand, OMNeT++ is gaining in popularity due to its powerful GUI and supporting hierarchical modelling. However, it has a gap in implementing one's own AQM algorithm, which is considered as a difficult task due to the complex structure and lack of supporting materials. OPNET Modeler is a very powerful tool and a good choice for complex scenarios, making it popular in the industrial world. Lucio et al. [131], in their comparison of NS-2 and OPNET, stated that both simulators produce the same results and have the same performance from the technical point of view, but NS-2 is more attractive to academic researchers because of its ready availability compared to the very expensive OPNET [132]. According to A. Deepak [133], NS-3 has developed well so far but there is still lack in terms of developing and implementing new AQM scheme. Besides, the authors have proposed a programed model that help to evaluate AQM schemes and they suggested more improvement and development as future work. Based on the above information, NS-2 was chosen as the simulation tool for this research.

3.5.3 Experiment Setup

As mentioned earlier, this research is conducted using a simulation approach as a method to study and evaluate the performance for the proposed HFAQM algorithm. The simulation was conducted in NS-2 version 2.35 on Ubuntu version 14.04 LTS of the Linux operating system. Furthermore, all experimental scenarios are executed on a bottleneck topology (dumbbell), which had been studied and approved as suitable topology for testing and evaluating the performance of different types of AQM [36], [133], [134]. HFAQM will be tested under a single wired link that contains one router node in between two access points (AP) and connected wirelessly to 100 sources and

destinations on each side, as shown in Figure 3.9. In order to achieve the research objectives, three different scenarios are run and analyzed accordingly, each scenario will have its own setup and configurations depending on its own simulation parameters specified below with the remaining shared simulation parameters in Table 3.2.

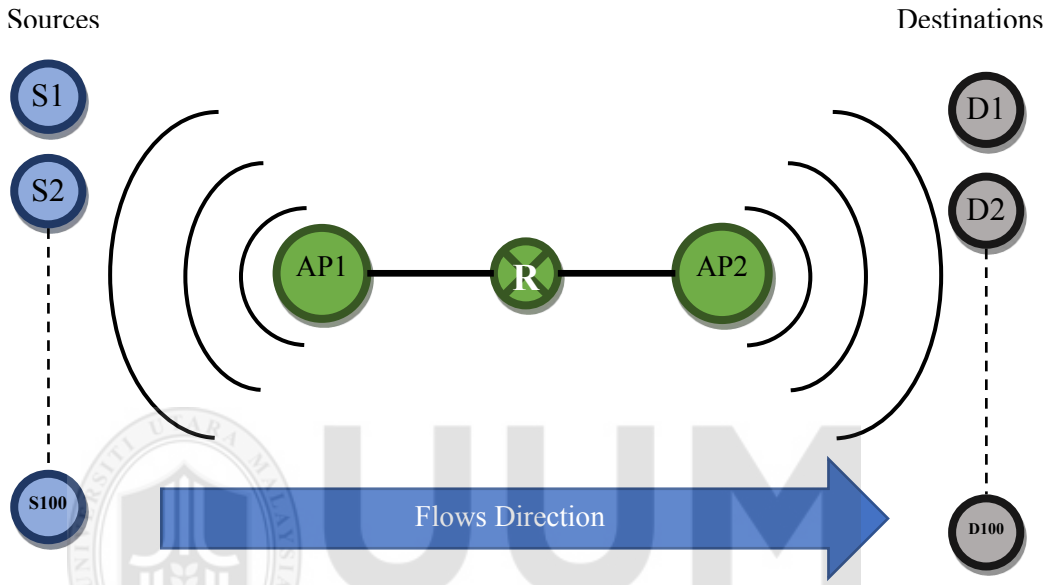

Figure 3.9. Single Link Bottleneck (Dumbbell) Topology

Table 3.2

Simulation Parameters

Parameters	Value
Simulation Topology	Dumbbell
Flow Type	Responsive (FTP over TCP-Reno) Unresponsive (CBR MP3 compress rate 180 Kbps) Short Flow (20 FTP packets over TCP-Reno)
Packet Size	1000 bytes
MAC Protocol	Ethernet 100 Mbps IEEE 802.11g
Queue Size	100 packets
RTT	60 ms Responsive flows 100 ms Short Flows
Simulation Time	100 seconds

Scenario 1: Three different types of flow are applied to the sources: first, 50 FTP flows transferred over TCP-Reno links with 1000 bytes packet size; second, 25 voice (MP3 compress rate 180 Kbps) flows implemented on CBR packets over UDP connections; and finally, 25 FTP flows with 20 packets for each link as short flows. This scenario is to study the performance of the proposed CIM and CFM and their effectiveness over different types of flow with consideration of fairness between all flows for 100 seconds at a time.

Scenario 2: In order to study stability HFAQM, the setting from scenario 1 is used. It starts with 29 FTP flows plus 4 UDP flows and 4 short flows. On the 25th second another 21 flows (7 flows from each type) will start to transfer their data through the single link; this event is repeated at the 50th and 75th seconds (every 25 seconds).

HFAQM will be evaluated by simulating the above two scenarios in NS-2.35. The proposed algorithm will be implemented in NS-2 and the results from the performance metrics compared with four other AQM schemes: RED, AVQ, REM and CoDel. RED is chosen because it was the first formal AQM, and has been implemented in some present-day real devices (such as: Alcatel-Luncent OmniAccess 5510/5740 unified services gateway; Brocade MLX Series – Multiservice IP/MPLS Routers; Cisco ASR 9000 System Aggregation Services Routers; HP MSR50 Series: and Juniper Networks M7i/ M10i Multiservice Edge Routers) [36]. As HFAQM is designed using the deterministic optimization approach, AVQ and REM have been chosen for the evaluation stage. CoDel, the newest AQM, has shown significant results in terms of fairness and stability in wireless environments; it is also the first AQM scheme to use queue-delay a as congestion indicator.

3.5.4 Performance Metrics

Different types of performance metrics have been used to evaluate AQM performance. According to this research prospective, the performance metrics are chosen to evaluate HFAQM can be divided into stability performance metrics and fairness performance metrics. In terms of stability, this research will use queue length and queue delay for evaluation as mentioned in [38, 45] as the most effective performance metrics to test queue stability. The Jain Fairness Index will also be used to measure fairness in the network [135]. For the general performance of the proposed algorithm, throughput, jitter and outgoing link utilization will be used.

Throughput: the total number of bytes received by the destination in the time unit (i.e. megabytes per second). The main aim behind any AQM algorithm is having a significant increase in throughput at the end node. In this research, throughput will be measured as the number of packets received successfully at the end node within the time unit (bytes per second), calculated as:

$$Throughput = \frac{N}{T} \tag{3.1}$$

where

N: The total number of bytes received

T: Time interval

Jitter: the variation in packet delay (latency) which is often used to measure the variability of end-to-end delay of packet delivery and can be calculated using the following mathematical formula:

$$jitter = abs(e2ed - \overline{e2ed}) \tag{3.2}$$

where

e2ed: End-to-end delay

$\overline{e2ed}$*: Previous end-to-end delay*

Outgoing link utilization (U): the amount of data carried by a link compared to the link's maximum capacity. In other words, link utilization measures the ratio of time the link consumes and can be calculated by:

$$U = \frac{BusyTime(second)}{SimulationTime(second)} \cdot 100\% \tag{3.3}$$

Queue length: the average of the packet occupied by the aggregation of the flows. It is an important metric to measure queue stability and can be calculated by:

$$\bar{Q} = \frac{1}{N}\sum_{i=1}^{N} Q_i \tag{3.4}$$

where

$\bar{Q}$: *Queue length*

Q_i: *The queue occupation by the link i*

N: The number of links

Queue Loss: the number of dropped packets in the congested buffer. It helps to measure stability in the queue which will reflect network stability overall. It can be easily measured by monitoring the buffer loss in NS-2.

Jain Fairness Index: the mathematical formulation to measure fairness among a number of links by calculating the received throughput for each link; it was originated by Raj Jain in 1984 [135] and is formulated as:

$$\mathcal{J}(x_1, x_2, \dots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot (\sum_{i=1}^{n} x_i^2)} \tag{3.6}$$

where

$\mathcal{J} = Fairness\ index$

$x_i: Throughput\ for\ the\ link\ i$

$n: Number\ of\ links$

3.6 Summary

Great details of this research have been given in this chapter to ensure that the objectives can be achieved precisely. For the first step (RC), the problem of fairness and stability was identified in Chapter 1. Whereas, the design approaches and configuration parameters were chosen in the second step (DS-1). The design of CIM and CFM formed and described in previous sections, with the verification and validation method for each mechanism, besides the implementation of each mechanism which had been conducted in PS. In the last step (DS-2) of DRM, the two mechanisms were combined to form HFAQM and evaluated in bottleneck scenarios in the NS-2 simulation tool using the prescribed performance metrics as mentioned in last step (DS-2) of DRM. After describing the methodology and experimental tool that was used to design and implement HFAQM in this chapter, the components of the proposed scheme (CIM and CFM) is presented in the next chapter with the design issues, mathematical modeling, validation and verification of each mechanism.

CHAPTER FOUR
HFAQM DESIGN AND IMPLEMENTATION

After establishing the research methodology in Chapter Three as a guideline to achieve the objectives of this research, this chapter proposes a new hybrid fair active queue management approach (HFAQM) for Wireless LAN networks. The proposed HFAQM will comprise two main mechanisms: congestion indicator mechanism and control function mechanism with embedded feedback mechanism. Each is designed, explained and validated in this chapter.

The congestion indicator mechanism is introduced in the next section, with the mathematical modelling, design, implementation and validation described in sub-sections. The control function mechanism and embedded feedback is covered in Section 4.3

4.1 Congestion Indicator Mechanism

The Congestion Indicator Mechanism (CIM) is one of the important mechanism in AQM schemes. CIM is responsible for indicating congestion before it happens. Many parameters are used to make this prediction, such as queue length, arrival rate, packet loss and any combination of these.

The tuning of these parameters in AQM schemes is very important as they can affect the stability of the mechanism and the speed of indication to reflect on the overall network. As discussed in Chapter Two, each parameter has its strength and weakness and how it achieves certain goals and objectives in different types of network.

The most popular types of CIM described in the literature are rate-based, queue-based and delay-based. The main goal of the rate-based congestion indicator mechanism is to maintain the arrival rate at the buffer at the target value, by controlling the arrival rate affecting the number of packets that can enter the buffer and therefore indirectly controlling the queue length. The queue-based congestion indicator mechanism aims to keep the queue length at the targeted value; some indicators considered the immediate value of the queue length but others take the average value.

However, considering the queue length as a congestion indicator leads to two major scenarios. First, when the queue length is large, the flows will suffer from a high probability of dropping, even though the arrival rate at that time might be quite low. For example, in Figure 4.1 the arrival rate *X(t)* is far lower than the drain rate *C(t)* and the queue is still draining rapidly, but the congestion indicator still indicates a high queue length ($Q_l > threshold$); the only solution in this scenario is by dropping packets and dequeuing others. This results in unnecessary packet loss which affects link utilization. In the second scenario, the queue is small and the congestion indicator mechanism calculates a lower probability of dropping than if the arrival rate was high. For example, a small buffer (*B*) with a very high arrival rate leads to a lower dropping fraction than it should be. Consequently, the queue-based congestion indicator mechanism is inept at detecting incipient congestion.

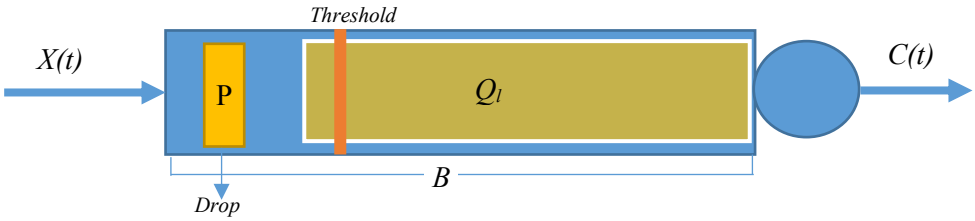

Figure 4.1. Simple Buffer Model

Although the queue-based congestion indicator mechanism controls the queue length by maintaining the packets in the buffer at the target value (*threshold*), this value is still non-zero. Therefore, in some cases the targeted queue length can be configured in the queue-based congestion indicator. This value is suitable for some specific network scenarios but may be inappropriate for others. How close the target is to the actual situation is highly load dependent. Additionally, this non-zero target value can cause a non-zero queueing delay. Therefore, using queueing delay instead of queue length can give a more meaningful measure for end node application. Furthermore, queueing delay by itself is independent of the link capacity and can be used to achieve high link utilization.

A congestion indicator mechanism that depends on one parameter to detect congestion can lead to instability and unfairness. In other words, a congestion indicator mechanism that uses only rate information can penalize some flows with a high transmission rate, achieving fairness but possibly leading to instability. Thus, queue length is still an important measure (although insufficient on its own). The combination of rate information and queue length in a single indicator can provide a tradeoff between fairness, stability and responsiveness in the AQM scheme overall.

For all these reasons, the proposed mechanism has hybridized the queue delay with the arrival rate to gate less tradeoff between fairness and stability. There are two advantages in using queue delay in the CIM. First, queue delay is a more meaningful measure than queue length so that the source can adjust their delay based on the queue delay (for example the RTT calculation in TCP). Thus, the transmission protocol considers a perfect measure parameter in calculating its delay time. Second, using the queue delay in CIM avoids the effect of different RTT flows, so all flows will be

treated the same. As a result, queue delay can have a direct effect on fairness in the overall AQM scheme. Therefore, equalizing the queue delay of different RTT flows will give an equal share of the output link capacity. For example, networks have different types of flow (responsive TCP flows and short TCP flows) transferring through a single link; by default, short flows will have a higher RTT than responsive flows. The CIM tries to equalize the delay of short and responsive flows by giving them a low indication value. In this way, responsive flows are penalized until they reach a suitable delay that is almost equal to the time that short flows take in the buffer. Thus, the RTT of responsive flows will be directly affected due to the penalized dropping.

The main objective of the proposed mechanism is to gain a higher fairness ratio by maintaining the delay time in the queue, and the optimization design approach has been adopted to achieve this. The next section discusses the design of the proposed CIM in solving the optimization problem by formulating the mathematical model. The validation of the model and mechanism are also discussed.

4.1.1 Congestion Indicator Mathematical Model

This section proposes the mathematical model for CIM over wireless LAN networks. The model is based on the optimization design approach and aims to avoid congestion with a better indicator equation and fairness among different types of flow.

Let us assumed a network with a set of links $L = \{1, \dots, L\}$ each link with finite capacity $c_l, l \in L$. These links are shared by a set of sources $S = \{1, \dots, S\}$, each source uses a set of links L_s that included in L. Thus, the set of sources that shares the same

link l is denoted as $s_l = \{s \in S | l \in L\}$. R_{ls} is a routing matrix defined by the sets of L_s as LxS.

$$R_{ls} = \begin{cases} 1, & if\ l \in L \\ 0, & otherwise \end{cases} \quad (4.1)$$

The transmission rate $x_s(t)$ is associated with each source s at a time t in packets per second, where $x(t) = [x_1(t), \ldots., x_s(t)]^T$ is the vector of all sources data rate. Associated with each link l is a congestion scaler called "price" and it is measured by per unit bandwidth denoted as $p_l(t)$ and let $p = [p_l(t), \ldots, p_l(t)]^T$ be the vector of all links' prices at a time t. Furthermore, $y_l(t) = \sum_s R_{ls} x_s(t)$ is the aggregate sources data rate at the link l and $q_s(t) = \sum_s R_{ls} p_l(t)$ is the end-to-end congestion measure for source s. Thus, the vectors of $y(t)$ and $q(t)$ will be $y(t) = Rx(t)$ and $q(t) = R^T p(t)$ where T denotes transposition.

Therefore, $x(t) = (x_s(t), s \in S)$ and $q(t) = (q_s(t), s \in S)$ are non-negative real values that depend on the source, $y(t) = (y_l(t), l \in L)$ and $p(t) = (p_l(t), l \in L)$ are non-negative real values that depend on the link. In other words, source s can control its own rate $x_s(t)$ which has a direct effect on the end-to-end congestion measure $q_s(t)$ of its path, but it will never observe the vector $x(t)$ or $p(t)$. On the other hand, link l can only observe the flow rate $y_l(t)$ and local congestion $p_l(t)$.

The source s can adjust its rate $x_s(t)$ for each period based only on $x_s(t)$ and $q_s(t)$ according to F_s function. Thus, the next source rate will be calculated based on the current source rate and end-to-end congestion measure as:

$$x_s(t+1) = F_s(x_s(t), q_s(t)) \quad (4.2)$$

Furthermore, the link congestion measure $p_l(t)$ is adjusted according to G_l and H_l functions based only on $y_l(t)$, $p_l(t)$ and possibly some internal variable $v_l(t)$, such as queue length at link *l*. Thus, the next congestion measure of link will be calculated based on the current congestion measure, flow rate and internal variable as:

$$p_l(t+1) = G_l\big(y_l(t), p_l(t), v_l(t)\big) \tag{4.3}$$

$$v_l(t+1) = H_l\big(y_l(t), p_l(t), v_l(t)\big) \tag{4.4}$$

Basically, F_s is the TCP model function that can control the transmission rate of the source. G_l and H_l are the AQM model which are implemented at the router where the buffer can be controlled. G_l is the probability function and H_l is the congestion indication measure, both functions can vary among different AQMs (this section will focus on H_l only).

Assuming the functions (Equations 4.2-4) have a set of equilibria (*x, p*). The implicit relation in the equilibrium between the end-to-end congestion measure q_s and the source rate x_s is defined by a fixed point (1) as $x_s = F_s(x_s, q_s)$. Furthermore, from the open set $A = \{(x_s, q_s) | x_s > 0, q_s > 0\}$, we can assume that F_s is continuously differentiable and $\partial F_s / \partial q_s \neq 0$. Then, the finding of a unique continuously differentiable function f_s from $\{x_s > 0\}$ to $\{q_s > 0\}$ can be modeled by using the implicit function theorem, such that $q_s = f_s(x_s) > 0$ around a fixed point. The definition of $f_s(0)$ will help to extend the mapping between x_s and q_s. Thus:

$$f_s(0) = \inf\{q_s \geq 0 | F_s(0, q_s) = 0\} \tag{4.5}$$

possibly infinity. Besides, $F_s(x_s, 0) = x_s$ if $(x_s, 0)$ is an equilibrium point. Then define $f_s(x_s) = 0$.

The definition of the utility function for all sources *s* is modelled as

$$U_s(x_s) = \int f_s(x_s)dx_s, \qquad x_s \geq 0 \tag{4.6}$$

that is unique up to constant.

U_s is a nondecreasing continuous function based on $f_s(x_s) = q_s \geq 0$ for all x_s and being an integral. U_s is concave as well because f_s is a non-increasing function, since maintaining severe congestion needs to reduce the source rate. Whenever f_s is decreasing strictly then U_s is strictly concave since $U_s''(x_s) < 0$. Moreover, a higher source rate yields a higher link utility, therefore, increasing the utility function indicates a greedy source and concavity implies fading in return.

Each source *s* that transmits data at a rate x_s observes a utility $U_s(x_s)$. Assuming the utility function is increasing due to concavity behaviour with respect to $x_s(t)$. Thus, the utility functions of all sources are cumulative $\sum_s U_s(x_s)$, then the primal problem becomes

$$\max_{x \geq 0} \sum_s U_s(x_s) \tag{4.7}$$

$$subject\ to \sum_s R_{ls} x_s(t) \leq c_l \qquad l = 1, \dots, L \tag{4.8}$$

The constraint is showing that the aggregate source rates x_s at link *l* should not exceed the link's capacity c_l. As mentioned, the objective function, called the primal optimal

solution, is strictly concave. The feasible solution set is compact since the function is continuous. However, the objective function is coupled with constraint by the sources rate x_s, solving the primal problem requires coordination among all sources rate which its impractical in real network. The best solution for this problem is by taking its dual as suggested by [26].

The first step in finding the dual problem from the primal problem is defining the Lagrangian as

$$L(x,p) = \sum_s U_s(x_s) - \sum_s p_l \left(\sum_{s \in S(l)} x_s - c_l \right)$$

$$= \left(\sum_s U_s(x_s) - x_s \sum_{l \in L(s)} p_l \right) + \sum_l p_l c_l \qquad (4.9)$$

Since the first term is separable in x_s and $\max_{x_s} \sum_s (U_s(x_s) - x_s \sum_{l \in L(s)} p_l) = \sum_s \max_{x_s} (U_s(x_s) - x_s \sum_{l \in L(s)} p_l)$, the dual objective function is [122, section 3.4.2], [137]:

$$D(p) = \max_{x_s \in l_s} L(x,p) = \sum_l B_s(p^s) + \sum_l p_l c_l \qquad (4.10)$$

where

$$B_s(p^s) = \max_{x_s \in l_s} U_s(x_s) - x_s p^s \qquad (4.11)$$

$$p^s = \sum_{l \in L(s)} p_l \qquad (4.12)$$

and the dual problem is:

$$\boldsymbol{D}\colon \min_{p\geq 0} D(p) \tag{4.13}$$

$D(p)$ from the first term is decomposed into S and separable into sub problems (Equations 4.11, 4.12). Since p_l is interpreted as the price at link l per bandwidth unit then p^s is the aggregation price for all links in the path of s per bandwidth unit which represents all congestion information that source s needs. Therefore, $x_s p^s$ is the bandwidth price for source s when it is transmitting at x_s, and $B_s(p^s)$ is the maximum utilization that source s can achieve at the price aggregation p^s.

By using the gradient projection method [136], [137], the dual problem will be solved where link prices are positioned in the opposite direction to the gradient $\nabla D(p)$:

$$p_l(t+1) = [p_l(t) + \gamma \frac{\partial D}{\partial p_l}(p(t))]^+ \tag{4.14}$$

where $\gamma > 0$ is a step size, and $[z]^+ = \max\{z, 0\}$. Form (3), $x_s(p)$ is the unique maximizer. Then;

$$D(p) = \sum_s \big(U_s\big(x_s(p)\big) - x_s(p)p^s\big) + \sum_l p_l c_l \tag{4.15}$$

As already mentioned, U_s is strictly concave. Thus, $D(p)$ is continuously differentiable with the derivative given by ([136], p. 669) as

$$\frac{\partial D}{\partial p_l}(p) = c_l - x^l(p) \tag{4.16}$$

where $x^l(p) \coloneqq \sum_{s \in S(l)} x_s(p)$ represents the aggregate sources rate at link l which is equal to $y_l(t)$. The price adjustment rule can be obtained by substituting Equation 4.16 into Equation 4.14 formulated for all links $l \in L$ as:

$$p_l(t+1) = [p_l(t) + \gamma(y_l(t) - c_l)]^+ \tag{4.17}$$

Assuming each link has a specific queue is very reasonable in order to control different characteristics of the sources (responsive and nonresponsive). This queue has a non-negative length and varies with time. Thus, let $q_l(t)$ be the queue length of the link. Since, $q_l(t)$ is non-negative, it has a non-negative queueing delay represented by $d_l(t)$. The queue delay is a very important variable in measuring congestion, which can be considered as $v_l(t)$ in Equation 4.4. $d_l(t)$ can be measured easily by subtracting the departure time (dequeue) from the arrival time (enqueue) as:

$$d_l(t) = dequeue_t - enqueue_t \tag{4.18}$$

By adding the $d_l(t)$ to Equation 4.16 gives:

$$\frac{\partial D}{\partial p_l}(p) = d_l(t) + y_l(t) - c_l \tag{4.19}$$

The new price adjustment rule can be obtained by substituting (4.19) into (4.14) again formulated for all links as:

$$p_l(t+1) = [p_l(t) + \gamma(d_l(t) + y_l(t) - c_l)]^+ \tag{4.20}$$

Equation 4.20 above is indeed consistent with H_l. The price increases if the aggregation sources rate at link l increases, weighted by γ which is a non-zero

smoothing constant and decreases otherwise. In equilibrium, the price stabilizes when $d_l(t) = 0$ (no queue delay) and the input rate equals the link capacity $y_l(t) = c_l$. However, this equilibrium does not mean fairness in the link. Some sources might have more space than others. In other words, $x_1(t)$ can be greater than $x_2(t)$ at a period of *t* which is not achieving fairness (the main objective of CIM).

In order to achieve fairness in the link, $d_l(t)$ will be used in the weighted summation. Let $d_l^*(t)$ be the equilibrium queue delay that satisfies fairness in the network. When the sources try to reach the same delay as at the queue, the sources will have the same share of the link. Thus, $d_l(t)$ in the weighted sum $(d_l(t) + y_l(t) - c_l)$ can be replaced with $((d_s(t) - d_l^*(t)) + y_l(t) - c_l)$. Therefore, source *s* with a queue delay higher than $d_l^*(t)$ will be penalized even if $y_l(t)$ does not exceed link capacity c_l thus forcing source *s* to reduce its $x_s(t)$ until it reaches its fair share of the bandwidth. Thus, Equation 4.20 can be formulated as:

$$p_l(t+1) = [p_l(t) + \gamma((d_s(t) - d_l^*(t)) + y_l(t) - c_l)]^+ \quad (4.21)$$

This suggests treating the queue delay as a fairness parameter by equalizing all sources' delay with the unique targeted delay, where, the aggregate sources rate is treated as congestion indicator parameters. However, both parameters will increase the price and be treated as the same overall. Moreover, the price increases if the weighted sum of rate and queue delay mismatches multiplied by γ is positive, and decreases if the weighted sum mismatch is negative. In other words, if the source delay is greater than the targeted delay, this means it is taking a greater share of the link than the remaining sources and will be punished even if the link is not congested, leading to the main objective of CIM which is fairness.

4.1.2 Design of CIM

The congestion indicator is the mechanism that measures congestion in the queue and tries to avoid congestion based on a mathematical formula. The mathematical formula might have one or multiple parameters to measure the congestion. CIM might have other objectives beside indicating congestion collapse before it happens. Therefore, CIM used a hybrid parameter in its mathematical formula. This hybridization enables CIM to achieve fairness in addition to indicating congestion.

From the previous section, Equation 4.21 is used as a congestion measure in CIM, enabling CIM to indicate the congestion level. However, $d_l^*(t)$ still needs to be defined due to its importance in the CIM designing purpose. As already mentioned, $d_l^*(t)$ is the targeted queue delay that source data packets should not exceed when travelling through the link queue. For the purpose of CIM, $d_l^*(t)$ value will be adaptive based on the link condition. Initially, the $d_l^*(t)$ value will be equal to $d_s(t)$ to enable CIM to indicate the congestion based on sources' rate $y_l(t)$ only. $d_s(t)$, on the other hand, will be calculated as shown in Equation 4.18.

Where $dequeue_t$ and $enqueue_t$ can be captured by adding a time stamp to each data packet arriving in the queue, and subtracting the time stamp record with the departure time. This estimates the exact time that the data packet took to leave the queue, as illustrated in Figure 4.2.

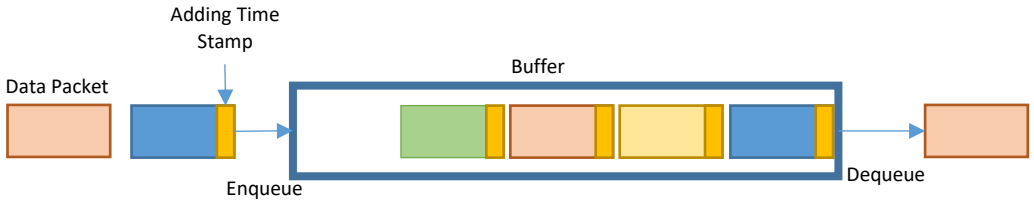

Figure 4.2. Packet Queueing Time

$d_l^*(t)$ is updated when the queue delay continues to increase and the queue length reaches more than its half. In other words, the queue delay increases when the queue is shared by more than one source. As the link-sharing sources increase, the data packet takes more time in the queue. The queue length can also give a good indication of the number of shared sources in the link. When this number increases, the queue increases dramatically especially when the link is shared by one or more non-responsive sources. Both queue delay and queue length are used to indicate that the link is shared by more than one source and that it is time to apply fairness among the sources, which means time to update $d_l^*(t)$ based on the equation below.

$$d_l^*(t+1) = d_l^*(t) + \frac{d_s(t) - d_l^*(t)}{2} \tag{4.22}$$

When the $d_s(t)$ is greater than $d_l^*(t)$, half the difference between $d_s(t)$ and $d_l^*(t)$ is added to the new target $d_l^*(t+1)$. However, when the $d_s(t)$ is less than $d_l^*(t)$, half of the difference is subtracted from the new target. This is repeated until the $d_s(t)$ becomes equal to $d_l^*(t)$, when the new target is not updated but keeps the same old value.

When the link becomes overflow, the $d_l^*(t)$ is updated for every time the queue suffers from drops or marking. Increasing the $d_l^*(t)$ enables CIM to penalize the nonresponsive sources more strictly, as the nonresponsive source has more data packets in the queue than the responsive ones. However, more packets in the queue means that the source has less delay time than the other sources. Thus, the weighted sum in Equation 4.21 is strictly different and the source needs to be penalized. The pseudo-code to illustrate the CIM steps is presented as follows:

Algorithm 4.1 UPON EACH PACKET ARRIVAL

Step 1. Adding time stamp to the packet

Step 2. **If** Queue Length >= Buffer/2 **Then** {

Step 3. Update $d_l^*(t)$ based on Equation (4.22)

Step 4. } **else** {

Step 5. $d_l^*(t) = d_l^*(t)$

Step 6. }

Step 7. Calculate $p_l(t)$ based on Equation (4.22)

Step 8. **If** $p_l(t)$<= 0 **Then** {

Step 9. $p_l(t) = 0$

Step 10. } **else** {

Step 11. Send $p_l(t)$ to Control Function to take an action

Step 12. }

4.1.3 Verification and Validation of CIM

The main reason for conducting verification is to ensure that the proposed CIM is implemented properly in the NS-2 simulation environment, and is programmed correctly in C++ programing language. As explained in Chapter Three, CIM was been verified and a snapshot of the implementation is shown in Figure 4.3 using the Eclipse platform. From the figure, it can be confirmed that CIM does not appear to contain bugs or errors and is programed correctly.

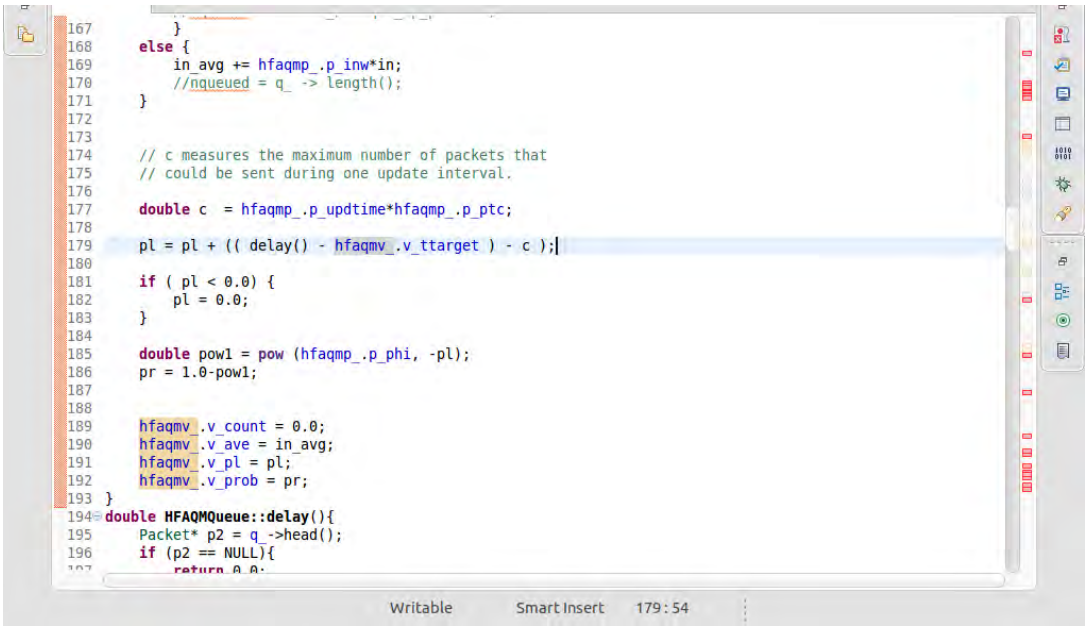

Figure 4.3. CIM Code in Eclipse

As already explained, the main requirement of CIM is to measure the congestion level periodically and accurately in order to avoid congestion in advance. The validation of CIM to ensure that it meets the requirements focuses on examining the queue length in the simulation scenario. The queue length of CIM is compared with the standard REM scheme because, like CIM, REM was designed using the optimization approach and a hybrid congestion indicator.

The topology of the simulation scenario is the dumbbell topology, conducted using the NS-2 simulator. Five wireless sources were used to send continuous data through a single wired link to the single wireless destination node. Two of the sources used UDP to represent non-responsive flows and the rest used TCP to represent a responsive one. The simulation was run for 50 seconds under the default simulation settings provided in Chapter Three. Figure 4.4 illustrates the simple dumbbell topology used in the validation of CIM.

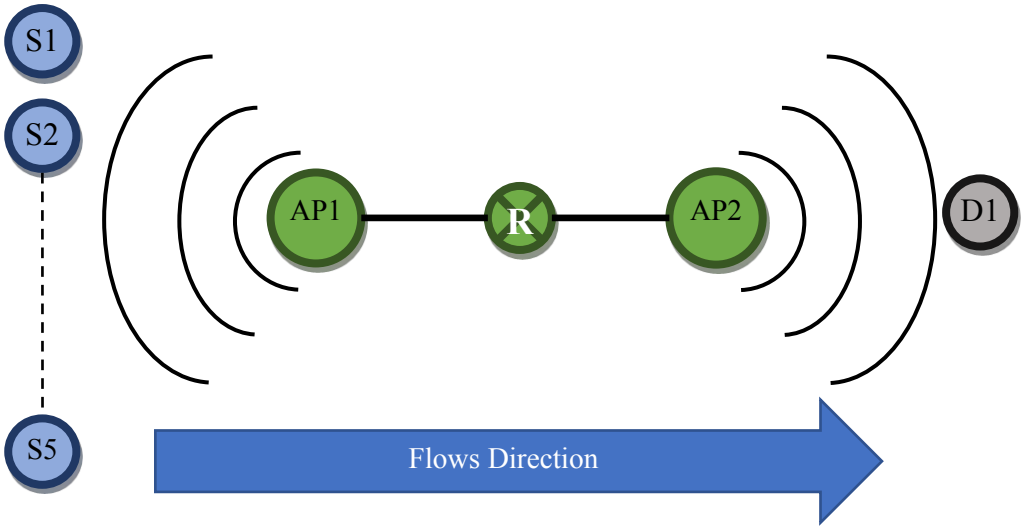

Figure 4.4. Dumbbell Topology with Five Sources

Exactly 1.5 seconds after the start of the simulation the TCP flows initiated transmission, and after 20 seconds the UDP sources began to send data through the link. At the 49th second, all sources stopped sending data through the link. The first 20 seconds represented CIM behaviour when the network was running in a low-load condition with responsive sources, while the last 30 seconds showed the effect of nonresponsive sources in the network on the behaviour of CIM.

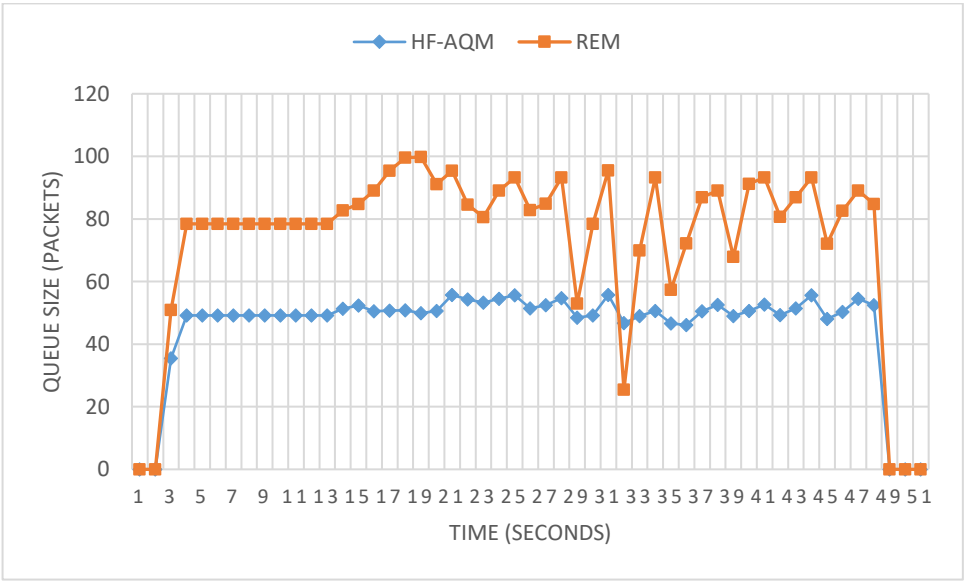

Figure 4.5. Validation Result (CIM vs REM Congestion Indicator)

The proposed CIM was validated over the dumbbell topology and the queue size was analyzed and is shown graphically in Figure 4.5. It is clear that CIM managed a lower queue length than REM even when combined with basic droptail as a control function, due to the price calculation that affected the queue delay and transmission rate as a congestion indicator. The queue size of REM also suffered from a high level of dropping especially after the UDP started trying to recover the network from the congestion that happened at 19 seconds. Moreover, CIM managed to keep the queue length as stable as possible without any high queue size or high queue loss by keeping the queue size between 55 and 35 packets.

After running the simulation and comparing the queue size of CIM and REM, the validation result showed that the proposed CIM achieved its purpose by anticipating congestion and keeping the queue size stable, even when working with the basic droptail control function.

4.2 Control Function Mechanism

The Control Function Mechanism (CFM) is responsible for marking or dropping packets according to the dropping probability function. This function maps the congestion indicator of the actively managed buffer to the dropping probability. Moreover, the probability obtained determines the dropping intensity of arriving packets at the current congestion level. The system model for AQM schemes is illustrated in Figure 4.6.

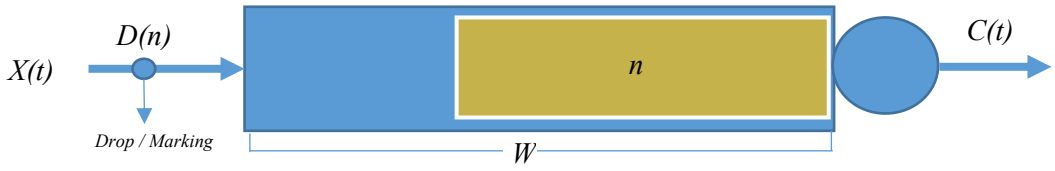

Figure 4.6. System Model for AQM

Where n is the chosen parameter for the congestion indicator (eg: queue length). The dropping probability function $D(n)$ is a non-decreasing function that maps a non-negative value $n \in [0, W]$ to [0,1], where W is a targeted positive integer (e.g. buffer size).

Among the infinite number of probability functions, three types of dropping probability function are popular in the literature and implementation. The first type is the exponential function, defined as:

$$D(n) = \begin{cases} 1 - \alpha^n, & 0 \leq n < W \\ 1 \quad , & n = W \end{cases} \tag{4.23}$$

The aggressiveness of this type depends inversely on the exponential function parameter represented by α which is a constant. The value of α is between zero and one $\alpha \in [0,1]$. When the value of α is closer to one, the exponential function is less aggressive in packet dropping, and vice versa. On the other hand, when the congestion parameter n becomes equal to the target W, the exponential function is given the one value which is the maximum aggressive point. Figure 4.7 illustrates the wave forms of the exponential function.

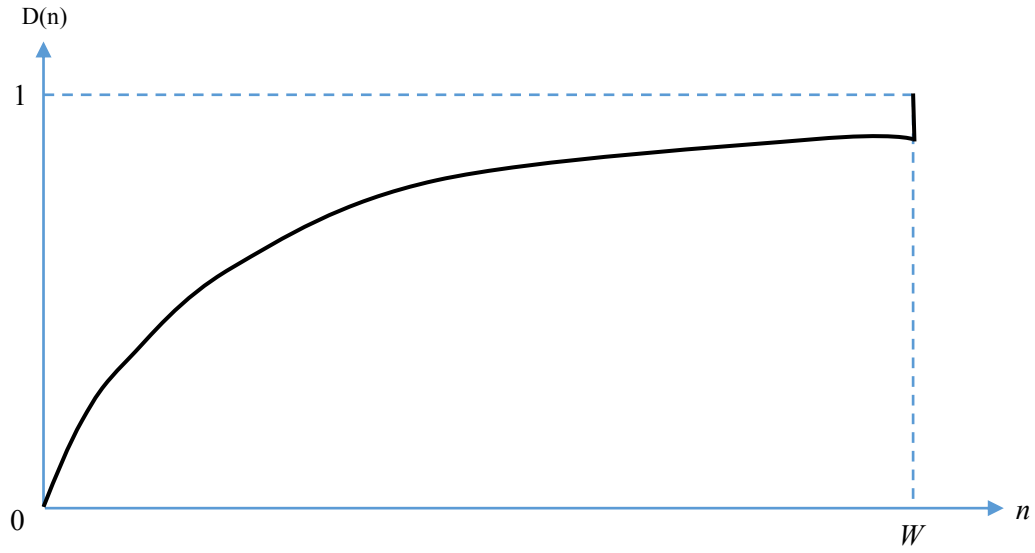

Figure 4.7. Exponential Probability Function

The second type of probability function is the simplest form, the *piece-wise linear function*, defined by triple parameters (max_{th}, min_{th} and max_p) as below:

$$D(n) = \begin{cases} 0, & 0 \le n < min_{th} \\ \frac{max_p(n - min_{th})}{max_{th} - min_{th}}, & min_{th} \le n < max_{th} \\ 1, & max_{th} \le n \le W \end{cases} \quad (4.24)$$

Piece-wise linear function is considered more complex than the exponential function because it depends on three function parameters: minimum threshold min_{th}, maximum threshold max_{th} and maximum probability max_p (the maximum value of probability before it reaches one). Upon packet arrival, if n is less than min_{th}, the packet is admitted to the queue. However, the packet is dropped if n is equal to or greater than max_{th}. When n is between min_{th} and max_{th}, the packet is dropped randomly, based on the linear function evaluated by n. The wave form of the piece-wise linear function is shown in Figure 4.8.

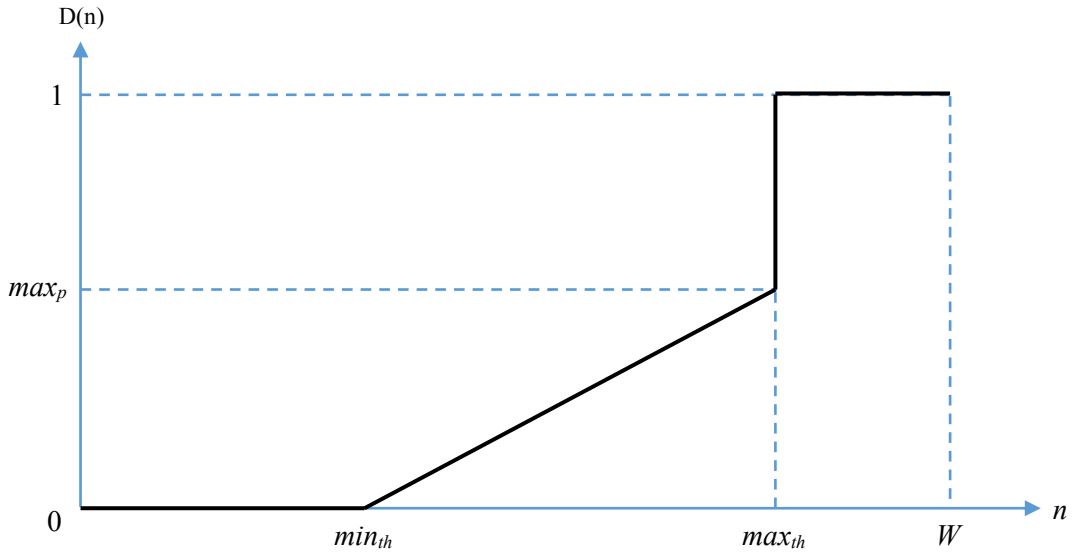

Figure 4.8. Piece-wise Linear Probability Function

The third type of probability function is the *stair-case function,* which is considered as the most complex type because it depends on an infinite number of coordination pairs as defined below:

$$\mathrm{D(t)} = \begin{cases} 0 & 0 \le n < w_0 \\ P_0 & w_0 \le n < w_1 \\ P_1 & w_1 \le n < w_2 \\ P_2 & w_2 \le n < w_3 \\ P_3 & w_3 \le n < W \end{cases} \tag{4.25}$$

The stair-case function is configured with an infinite number of probability conditions, starting from 0 until P_i where $i\epsilon\{0,1,\ldots,\infty\}$. Each probability condition is assigned to a coordination pair, for example P_0 is assigned to (w_0,w_1). This function has been implemented in Juniper Networks' router in which the user can define up to 64 probability condition and coordination pairs (default is 4). Figure 4.9 is the wave form of the stair-case function.

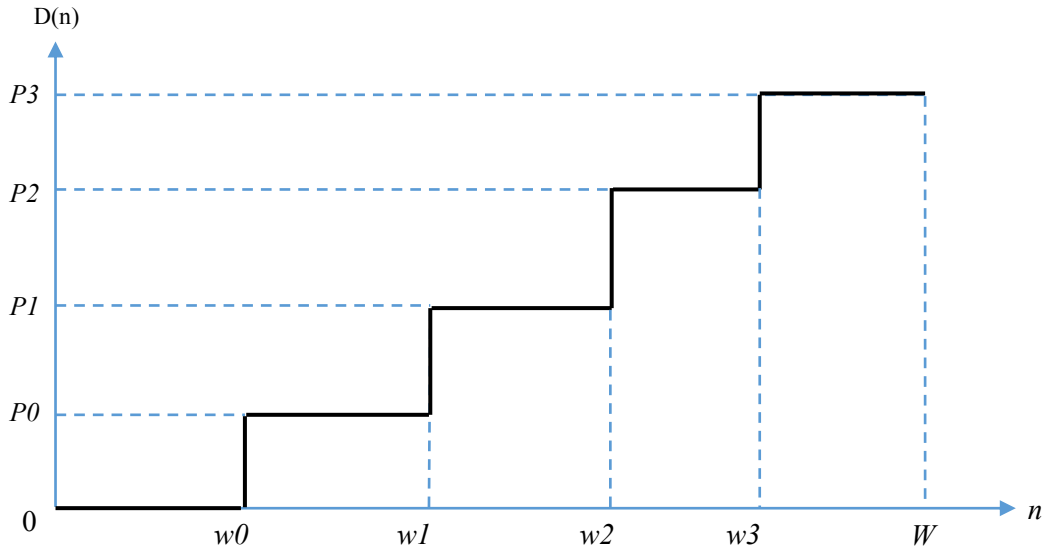

Figure 4.9. Stair-Case Probability Function

4.2.1 The Analytical Model of CFM

CIM defined the key parameters that should be included to avoid congestion as well as achieving fairness. AQM with CFM converges to accomplish efficient buffer management scheme. The main objective of CFM is to take the action of dropping or marking the packets that do not satisfy the condition of CIM, and to improve the stability in the network.

As mentioned above, the design behind the control function is using a suitable probability function. For the purpose of CFM, the exponential probability function was chosen in CFM for two reasons. First, exponential function does not start with a zero value, which means that it reflects and measures the probability based on the price given by the indicator mechanism even if the link is not yet congested. This is very important CIM design to achieve fairness as a second objective. Secondly, the exponential function is less complex than the other types.

The exponential function is defined in Equation 4.23, *n* is representing the congestion indicator parameters. However, CIM used two parameters in its price formulation, queue delay $d_s(t)$ and the aggregate sources rate $y_l(t)$. Therefore, CFM represents *n* as a function of $(d_l(t), y_l(t))$ so $n = f(d_l(t), y_l(t))$. Whereas, $p_l(t)$ is a function of $(d_l(t), y_l(t))$ as formulated in Section 4.2.1. Thus, $n = p_l(t)$. The exponential function of CFM is defined as:

$$D(n) = \begin{cases} 1 - \alpha^{-p_l(t)}, & d_s(t) < d_l^*(t) \text{ and } y_l(t) < c_l \\ 1 \quad , & d_s(t) = d_l^*(t) \text{ and } y_l(t) = c_l \end{cases} \tag{4.26}$$

From Equation 4.26, it can be seen that $p_l(t)$ is multiplied by -1, the purpose behind that is to increase the aggressiveness of the probability function by making the exponential value decrease when the price $p_l(t)$ increases. Also, the second condition in the equation can be included in first condition because when $d_s(t) = d_l^*(t)$ and $y_l(t) = c_l$ is the equilibrium point this means that $D(n)$ should equal to 0 and the whole system works perfectly. However, when $d_s(t) > d_l^*(t)$ and $y_l(t) > c_l$ the price $p_l(t)$ is very high. Thus, the value of the term $\alpha^{-p_l(t)}$ is close to the zero and can be ignored. Therefore, the probability function is formed as follow in all conditions:

$$D(n) = 1 - \alpha^{-p_l(t)} \tag{4.27}$$

4.2.2 Design of CFM

The key feature behind control function mechanism is to put the decision of the congestion indicator into action to drop or mark the packets by using a well defined probability function. Besides, CFM has another objective to achieve which is improving stability. The dropping and marking can be very harmful to the network if it is not decided wisely.

From Equation 4.27, α is a constant and its value should be $0 < \alpha < 1$. On the other hand, $d_l^*(t)$ is the targeted queue delay, representing the targeted time that the packet should not exceed when it travels through the buffer. $d_l^*(t)$ is measured in micro seconds which is logically less than one. Therefore, there is no single reason to replace α with $d_l^*(t)$ value. However, using $d_l^*(t)$ instead of α has two advantages:

- Increasing the stability by making CFM more adaptive to the network condition in each period of *t*.

- Decreasing the complexity of CFM by replacing an extra constant with a predefined parameter.

As explained in Section 4.2.2, $d_l^*(t)$ is not updated unless the queue length exceeds half the buffer size and its value was equal to $d_s(t)$ before that. Therefore, CFM should adopt this condition in its design. Thus, when $d_l^*(t)$ is equal to $d_s(t)$, CFM uses α in its probability function and $d_l^*(t)$ otherwise. Thus, the pseudo code of CFM is described as below.

Algorithm 4.2 IN EACH PERIOD OF TIME

Step 1. If ($d_s(t)$==$d_l^*(t)$) **Then** {

Step 2. Calculate $D(n)$ as $D(n) = 1 - \alpha^{-p_l(t)}$

Step 3. } **else** {

Step 4. Calculate $D(n)$ as $D(n) = 1 - d_l^*(t)^{-p_l(t)}$

Step 5. }

4.3 Verification and Validation of CFM

The verification of CFM was done using the method described in Chapter Three. A snapshot of CFM code is presented in Figure 4.10, confirming that CFM has been programed correctly and implemented without any apparent bugs or errors.

Figure 4.10. Some of CFM Code in Eclipse

CFM was validated to ensure that the probability function works correctly and adapts to the network status. The validation of CFM focused on examining the queue loss in the simulation scenario and compared it with the conventional REM control function which is used as a reference of performance bound.

The topology used in this validation was the same as the topology that is used for validating CIM (Section 4.2.3), although for the CFM validation purpose, the topology used 10 sources instead of five. Five of these sources used TCP as the transport protocol, representing responsive sources, and the remaining five used UDP to

represent nonresponsive sources. The simulation was run for 60 seconds using the NS-2 simulator. Figure 4.11 illustrates the validation topology.

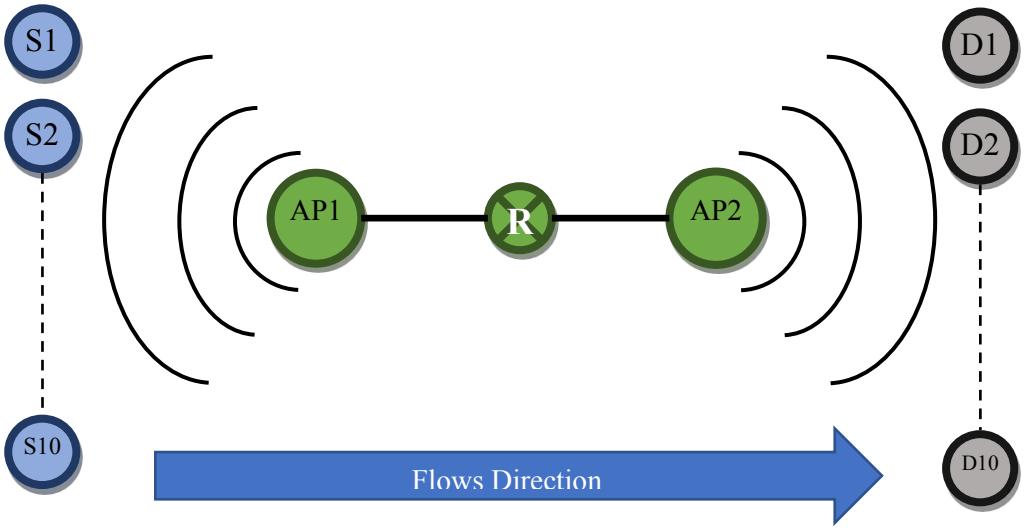

Figure 4.11. Dumbbell Topology for CFM Validation

For the validation purpose, CFM was designed to drop the penalized sources instead of marking them, even for the responsive sources, in order to examine the aggressiveness of the CFM probability function. At exactly 1.5 seconds into the simulation, all sources using TCP started to send their data through the link. These sources were kept running alone through the single link to examine the behaviour of CFM in the presence of responsive flows only. To examine the aggressiveness of the CFM probability function in the presence of nonresponsive flows, three of the UDP sources started sending data through the single link after 20 seconds into the simulation. At 40 seconds, the other two UDP sources started sending data to see how CFM adapted to the changing network.

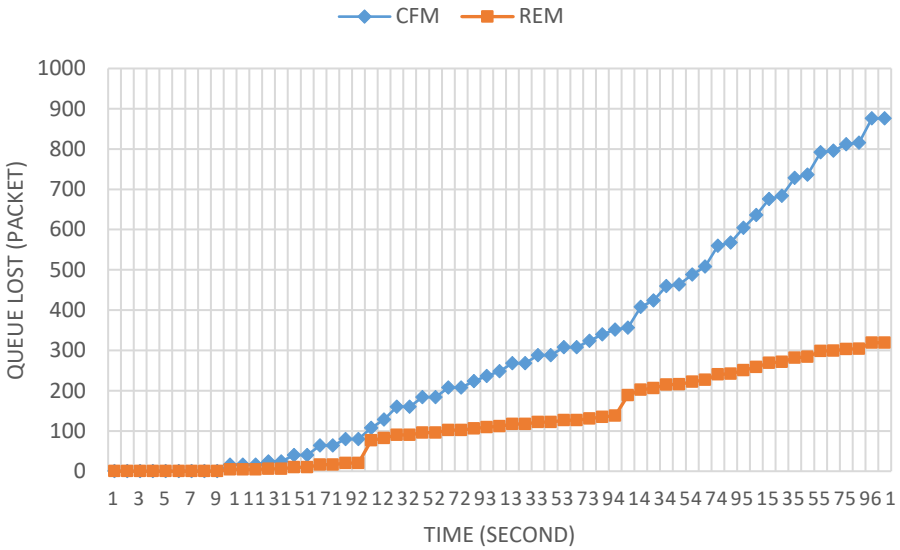

Figure 4.12. CFM Validation Result

The proposed CFM was validated over dumbbell topology and the queue lost was analyzed and graphed in Figure 4.12. The result shows that CFM has a very aggressive probability function due to the high queue loss. On the other hand, REM had lower queue loss because it used marking for the responsive flows. Moreover, it is noticeable that queue loss for CFM increased gradually throughout the simulation due to the adaptation of the probability function using the queue delay to control the dropping aggressiveness. REM exhibited a jumping queue loss whenever the flows or the load increased on the queue. From the validation result, it can be concluded that the proposed CFM has achieved its intended task with the aggressive probability function.

4.4 Summary

This chapter introduced congestion indicator mechanism CIM and control function mechanism CFM for HFAQM scheme. The proposed mechanisms is the main component of the HFAQM scheme for detecting and controlling the congestion in WLAN environment as described in this chapter together with mathematical model,

pseudo code, verification and validation of each one individually. The next chapter will describe the combination of both mechanism together to form HFAQM scheme. The evaluation process and the result discussion describe with greater detail in next chapter as well.

CHAPTER FIVE
HFAQM PERFORMANCE ANALYSIS

In the previous chapter, CIM and CFM were introduced to indicate and control congestion in the queue. Moreover, the proposed mechanisms have to be combined together to complete the action of full scheme. This chapter is proposing the combination of the proposed mechanisms with great detail on performance evaluation and results discussion.

5.1 HFAQM Overview

The primary objective of this research is to propose HFAQM for WLAN networks. HFAQM is a buffer management scheme that uses hybridization between queue delay and aggregate source rates to indicate and control congestion in the buffer. Moreover, the design is based on an optimization approach which features two mathematical formulae: price and probability function. Furthermore, HFAQM integrates the two proposed mechanisms, CIM and CFM, as described in Chapter Four, to improve AQM performance in terms of fairness and stability. Finally, HFAQM has made several changes on AQM to cope with the added objective. These changes are not necessarily optimal, but they are a good step towards improving AQM performance in WLAN networks.

The buffer in the network can travel through different states at various times: empty, nearly full and overflow. AQM scheme should treat each state differently to improve the performance of the network. HFAQM therefore controls three types of buffer state, empty, shared and overflow buffer, as shown in Figure 5.1 with the description of each state as flow.

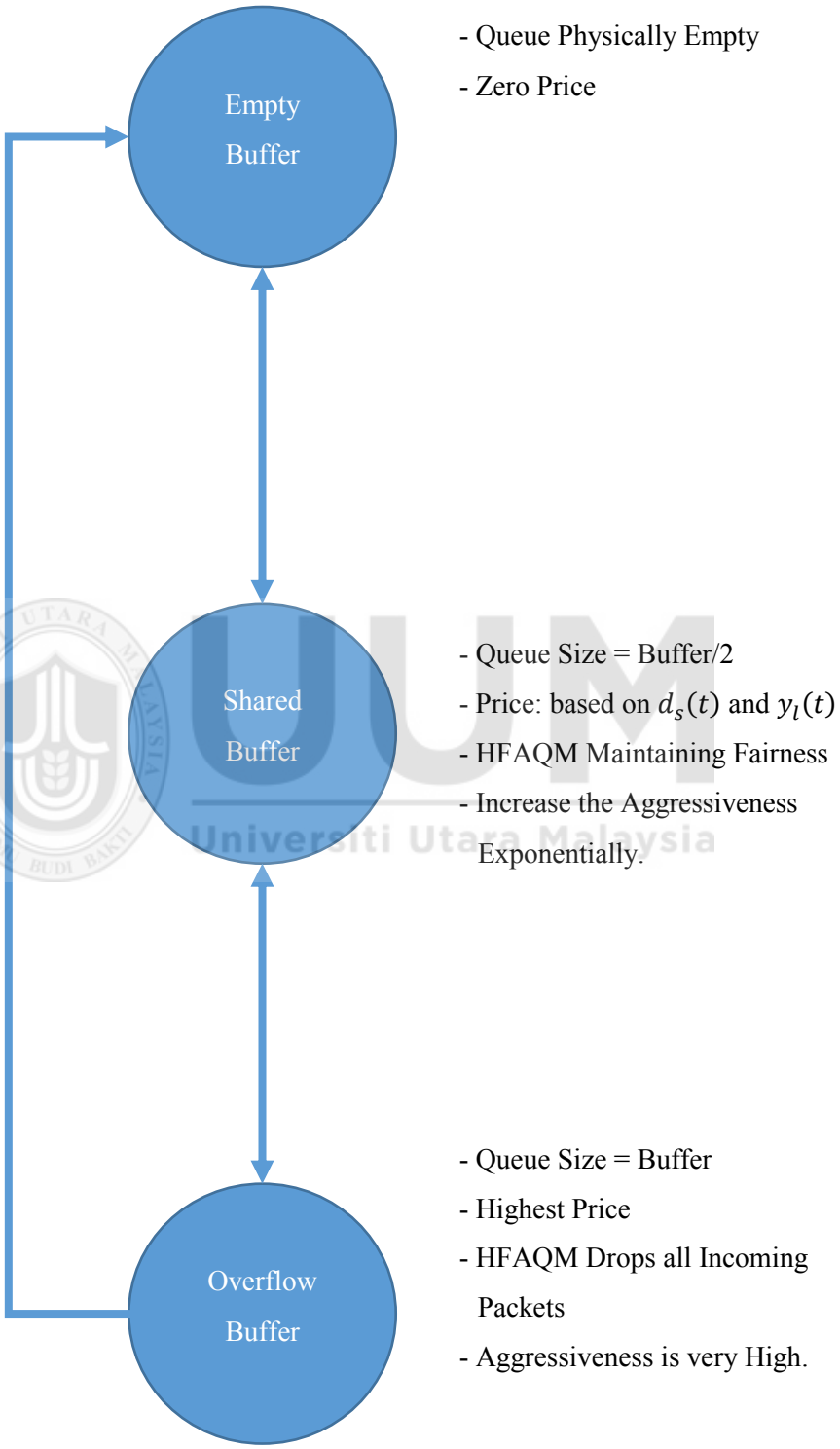

Figure 5.1. Types of Buffer States for HF-AQM Scheme

Empty Buffer:

It is the state when HFAQM treats the buffer like empty. This happened when the queue is physically empty or very low. In this state, HFAQM is working with zero price because the queue does not show any need to take any action for any reason.

Shared Buffer:

It is the state when the buffer shared by more than one source. HFAQM indicates this state when the queue size reaches half the buffer size. In this state, the proposed AQM scheme starts to maintain the rates of the sources to control fairness among them. The price is therefore calculated based on both queue delay and source rate, and the probability function starts to maintain the dropping based on calculated price. Moreover, the aggressiveness of HFAQM increases exponentially based on CFM.

Overflow Buffer:

It is the state when the buffer is full. This state usually occurs when the buffer is shared by nonresponsive sources. In this state, HFAQM adopts the most aggressive state because the price is at the highest value and the probability function is near to one. Therefore, HFAQM keeps penalizing all the nonresponsive flows and even the responsive ones until the buffer exits this state.

5.2 Combination of the Two Mechanisms

As outlined in Chapter One, several objectives can be achieved in designing a new AQM scheme in addition to avoid and control congestion, for example, low end-to-end delay and high link utilization. However, this section is providing a design of new

AQM scheme which is called HFAQM. This new scheme has the following features besides avoiding congestion proactively:

i. Improving fairness among different types of flow by combining queue delay with sources rate as designed in CIM

ii. Improving the stability of the system by making the proposed scheme more adaptive to network conditions as designed in CFM.

iii. Providing predictable queue delays by using queue delay as a parameter to penalize the different source types.

iv. Ensuring high link utilization by using the sources rate as a parameter to keep the sources rate as high as possible without exceeding link capacity.

In order to implement these features, CIM and CFM are combined in a single HFAQM scheme. This combination can be challenging in terms of mapping the parameters in NS-2 which uses two different programing languages (C++ and OTCL), and mapping the actions of HFAQM scheme in correct code procedures. However, HF-AQM scheme should contain the main three components as mentioned in Chapter One which are: congestion indicator, control function and feedback mechanism.

The first and second components of HFAQM scheme are CIM and CFM respectively. The third component, the feedback mechanism, is also important to complete the action of CIM and CFM. Therefore, HFAQM feedback mechanism is using both dropping and marking as an action in its design. When the link is shared by different types of responsive and nonresponsive flows, the feedback mechanism is used to drop

for the nonresponsive flows and marking for the responsive ones. When the buffer enters the overflow state, the feedback mechanism uses dropping as feedback for both responsive and nonresponsive flows and treats them in the same way to increase the aggressiveness of HFAQM. The pseudo-code of HFAQM feedback mechanism is presented as follows:

Algorithm 4.2 UPON EACH ENQUEUE OR DEQUEUE

Step 1. If $D(n)$ == 0 **Then** {
Step 2. Dequeue the packet
Step 3. } **else** {
Step 4. **If** Buffer Full **Then** {
Step 5. Drop the packet
Step 6. } **else** {
Step 7. **If** Packet ECN Capable **Then** {
Step 8. Mark the packet
Step 9. } **else** {
Step 10. Drop the packet
Step 11. }
Step 12. }
Step 13. }
Step 14. Dequeue the packet

HFAQM's three mechanisms are implemented in two different stages: enqueue and dequeue. Enqueue is the stage when the packet arrives at the buffer, and dequeue the stage when the packet leaves the queue. At each stage, HFAQM has different actions which are considered as follows:

Enqueue:

There is important action HFAQM should take upon each packet arrives in the queue,

HFAQM records the time of packet arrival. This action is done by adding time stamp to each packet, records the time when that packet enters the queue. This time is used to calculate the queue delay. HFAQM also measures the congestion level by calculating the price using the queue delay and aggregate sources rate as described under CIM design in Chapter Four. Moreover at this stage, if the buffer is in the empty state, CIM calculates the price based on aggregate sources rate only. When the buffer enters the shared or overflow state, CIM starts to calculate the price based on queue delay and sources rates, and HFAQM starts penalizing the sources based on CFM, using the feedback mechanism of either dropping or marking.

Dequeue:

HFAQM completes the calculation of queue delay at this stage by subtracting the leaving time from the recorded time stamp representing the arrival time. Moreover, the targeted time is set equal to the average queue delay when the buffer at empty state, and HFAQM works on the normal FIFO algorithm. However, the targeted queue delay is updated in this stage when the buffer enters the shared state. Also at this stage, when the buffer enters the overflow state, the feedback mechanism drops packets according to the CFM calculation to increase the HFAQM's aggressiveness and exit the congestion collapse as quickly as possible. Figure 5.2 illustrates the combination of HFAQM mechanisms.

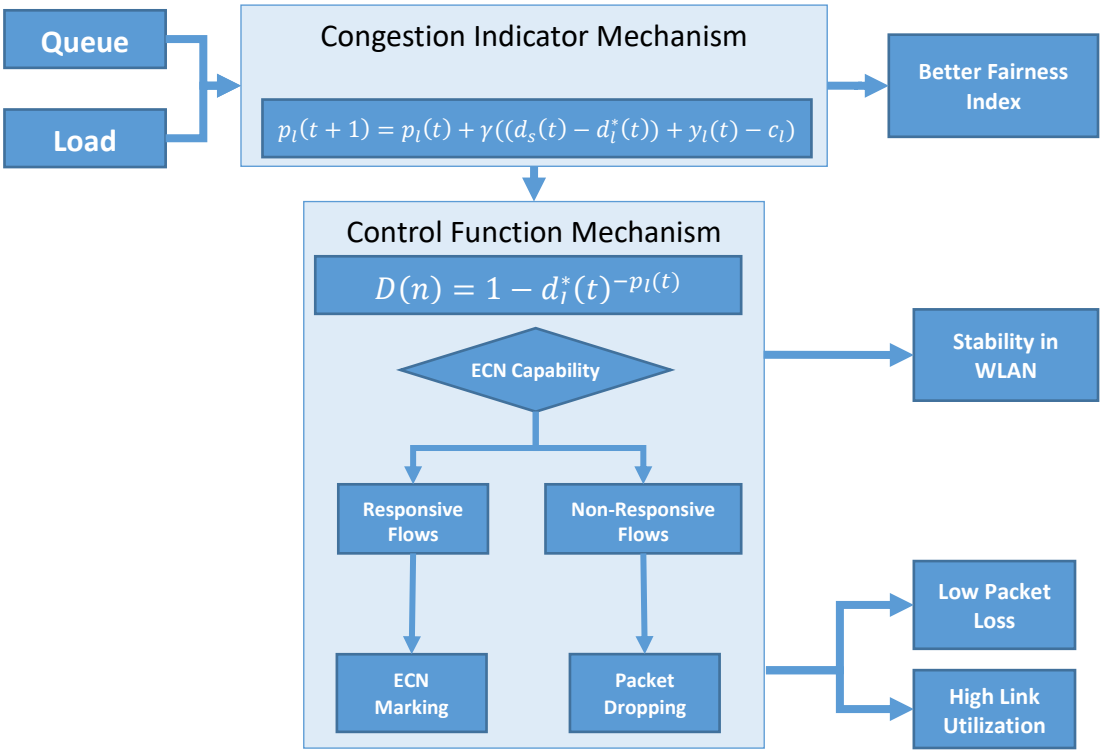

Figure 5.2. Combination of HFAQM Scheme Mechanisms

HFAQM was implemented in the NS-2.35 simulation environment based on REM code. The three components of HFAQM (CIM, CFM and the feedback mechanism) were combined and implemented in the enqueue and dequeue processes as mentioned previously. Incorporating CIM, CFM and the feedback mechanism aims to help HFAQM to take more accurate control and enhance AQM performance in terms of fairness and stability. Some of HFAQM source code is shown in the Figure 5.3.

```
187
188
189     hfaqmv_.v_count = 0.0;
190     hfaqmv_.v_ave = in_avg;
191     hfaqmv_.v_pl = pl;
192     hfaqmv_.v_prob = pr;
193 }
194 double HFAQMQueue::delay(){
195     Packet* p2 = q_->head();
196     if (p2 == NULL){
197         return 0.0;
198     }
199     else {
200     hdr_cmn* ch=hdr_cmn::access(p2);
201     double ent=ch->ts_;
202     double now=Scheduler::instance().clock();
203     return now-ent;
204 }
205 }
206 void HFAQMQueue::ttargetupdate()
207 {
208     double timetemp = delay();
209     if (timetemp > hfaqmv_.v_ttarget && !dmtrger){
210         hfaqmv_.v_ttarget=timetemp;
211     }
212 }
                                Writable    Smart Insert    185 : 44
```

Figure 5.3. Some of HF-AQM Code

5.3 Performance Evaluation of HFAQM

The main goal of performance evaluation stage is to test HFAQM's behaviour in congested WLAN networks. HFAQM was compared against four different AQM schemes: RED, REM, AVQ and CoDel. RED was chosen because it was the first formal AQM scheme and CoDel because it is the latest. Where, REM and AVQ were used in the comparison because they take the same design approach as HFAQM which is Optimization Approach.

As explained in Chapter Three, HFAQM was evaluated under two different scenarios, both scenarios used a single bottleneck represented by simple dumbbell topology. The first scenario was designed to test the fairness in the network by using the Jain Fairness Index as performance metrics. However, the second scenario was designed to evaluate network stability by measuring queue length and queue loss as the main metrics to test the HFAQM scheme in terms of stability. For the general performance evaluation,

other metrics were measured to gain a deeper understanding of HFAQM's behaviour. Among a variety of performance metrics, jitter, throughput and link utilization were used for general performance proposes.

5.3.1 Fairness Evaluation of HFAQM

It is very common to have bottleneck topology like in WLAN, especially when multiple wireless nodes are connected to a single access point, and multiple access points connected to a single router. This topology has been used widely to evaluate congestion control schemes and algorithms. Therefore, HFAQM was tested over a single dumbbell topology with 100 sources and 100 destinations connected through three access points and one router, as illustrated in Figure 3.9. The sources consist of three different types: 50 sources used a continuous TCP connection to represent responsive sources; 25 sources sent data through the UDP protocol to represent nonresponsive sources; and finally, 25 short flows represented sources that used TCP with fixed number of packets. The detailed information of the simulation model were listed in Table 3.2.

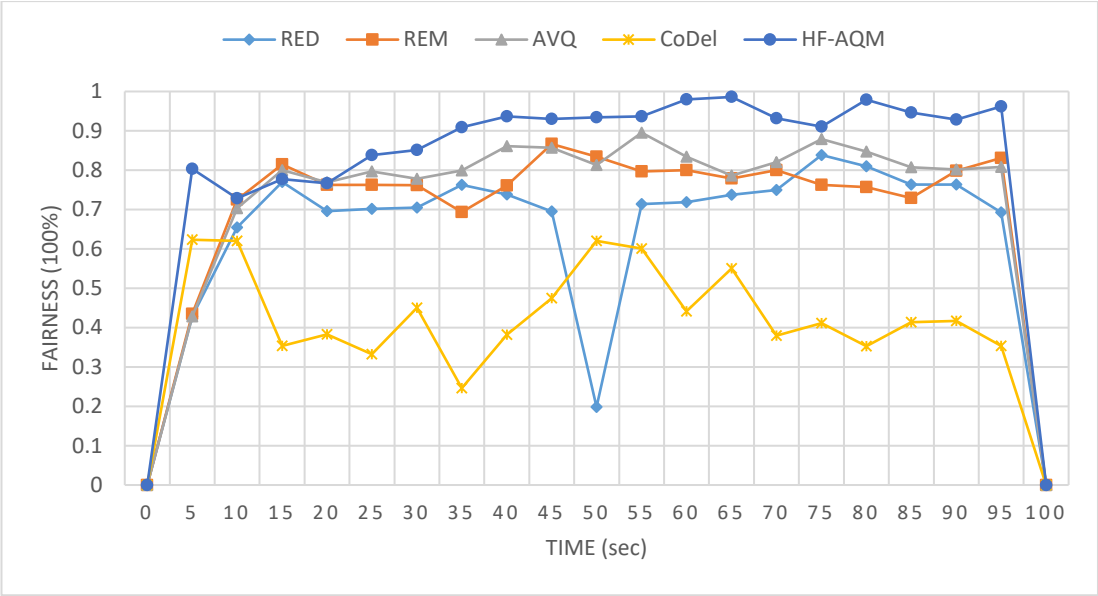

Figure 5.4. Fairness Results

Figure 5.4 shows the fairness of HFAQM together with the four comparative AQM schemes in a single dumbbell topology as mentioned previously. The Jain Fairness Index was used to measure fairness as mentioned in Chapter Three. All the AQM schemes had different levels of fairness, but it is noticeable that HFAQM and AVQ have the highest comparable values. The AVQ scheme has achieved a comparable fairness values which ranging between the lowest value which is 45% at 5th second and the highest value at 55th second which it reaches 90%. Moreover, REM has an approximately similar result to AVQ but it ranging between 45% and 87% at 5th and 55th second sequentially. On the other side, CoDel had the lowest fairness among the rest with unstable slope which it reached the 62% at 50th second and the lowest value is 24% at 35th second.

However, it is noticeable that all AQM schemes had a very low value on the start of the simulation around 5th second except for HFAQM which had the highest value which is 80%. Thus, HFAQM was able to apply fairness among different types of flow at the very early stage. Besides, HFAQM achieved the highest fairness levels among

all, it is rated between 72% at 10 seconds and 98% at 65 seconds. Therefore, HFAQM has achieved better fairness than RED, REM, AVQ and CoDel.

5.3.2 Stability Evaluation of HFAQM

As mentioned previously, the effect of HFAQM on buffer stability was tested. According to [36], the performance of AQM scheme should not dramatically changed with a sudden change in network condition. It is important for an AQM scheme to avoid oscillations between buffer underflows and overflows, which inevitably lead to link under-utilization and oscillating queue delay. The suggested stability test is to measure the queue length when the number of connections flowing through the queue suddenly increased. Therefore, this scenario used the same topology as in the previous section but with some small changes. The simulation started with 25 responsive sources, and the remainder were divided into three groups, each consisting of five sources from each type (responsive, nonresponsive and short flows). These three groups started sending data at second 25, 50, and 75 respectively, as shown in Figure 5.5.

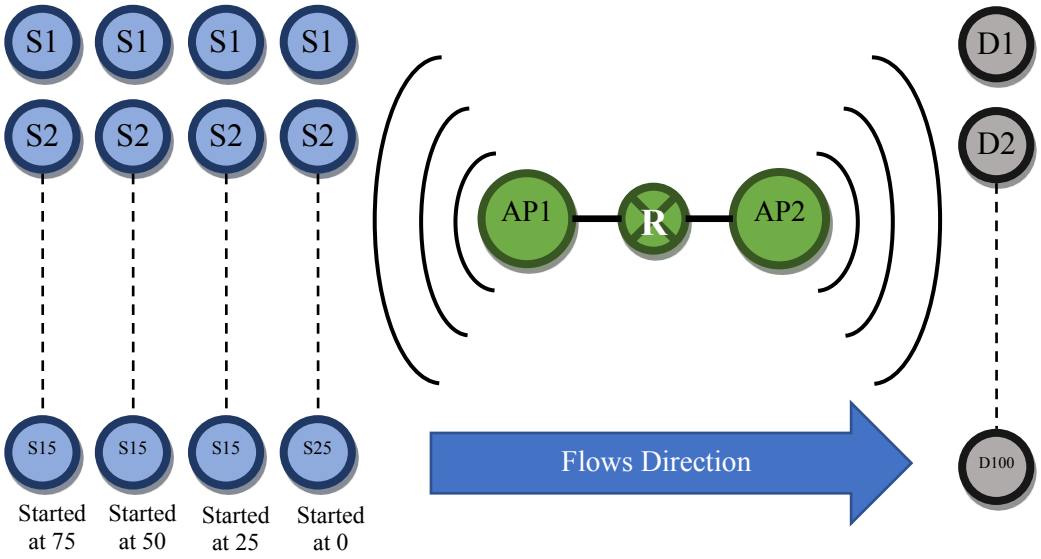

Figure 5.5. Dumbbell Topology for HFAQM Stability Test

The result of the simulation is shown in Figures 5.6-5.10. First, RED started with an average queue length and low queue loss; the lowest value was when the simulation reached 25 seconds. After this, the first group started to send data and the queue length made the first jump from 13 packets to 25 packets, and increasing from then on. When the simulation reaches 50 seconds, the queue length took the second jump from 38 to 71 packets with a dramatic increase in queue drop. When the last group started sending data, the queue length took the last jump from 50 to 80 packets with a corresponding jump in queue drops, as shown in Figure 5.6.

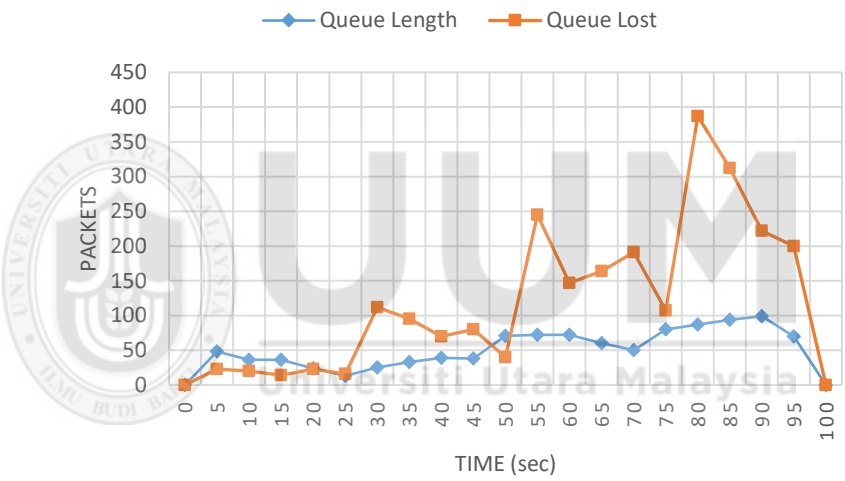

Figure 5.6. Queue Length and Queue Loss for RED

REM also exhibited unstable behaviour. It started with a low queue length and queue drop, but kept jumping from 30 to 50 packets, from 70 to 80 packets and from 74 to 98 packets at 25, 50 and 75 seconds respectively, along with increasing queue loss, as shown in Figure 5.7.

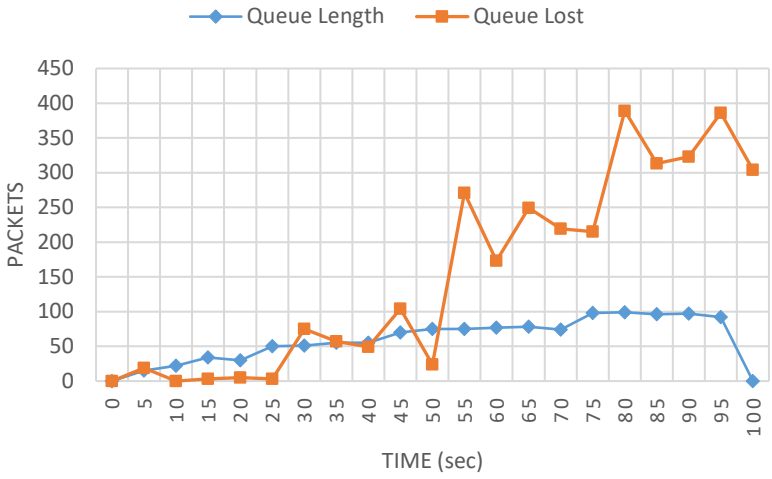

Figure 5.7. Queue Length and Queue Loss for REM

On the other hands, AVQ's unstable behaviour when the network condition changed is illustrated in Figure 5.8. It is easy to find the point at which queue length changed, together with increasing queue loss. As the number of sources increased, so did the queue length and queue loss.

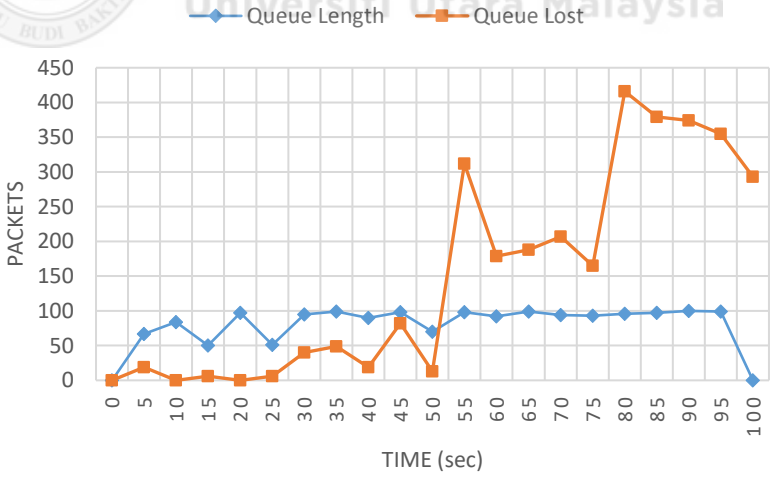

Figure 5.8. Queue Length and Queue Loss for AVQ

CoDel had an unexpectedly low queue length, ranging from 0 to 9 packets, but this came at the price of a very high queue loss, which increased dramatically, as shown in Figure 5.9.

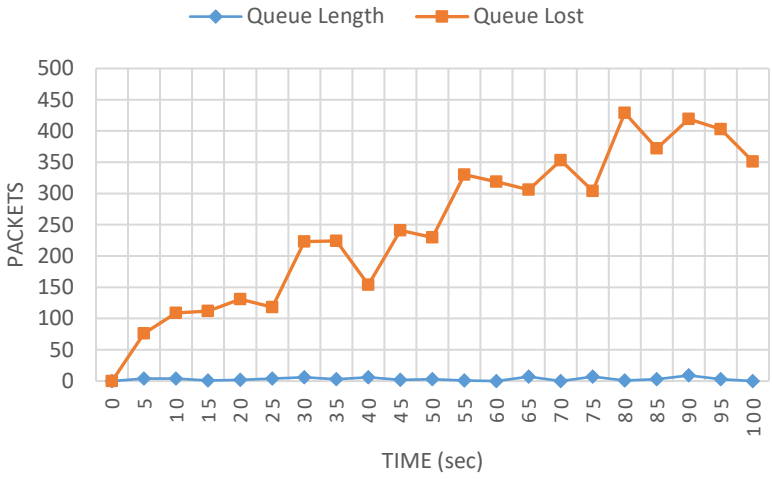

Figure 5.9. Queue Length and Queue Loss for CoDel

Lastly, HFAQM performed best of all, with a stable queue length starting at 55 packets and varying between 73 and 98 packets without any noticeable changes as the number of sources increased. Queue loss, however, increased cumulatively from 20 packets until it reaches to a maximum value of 240 packets, which was nevertheless lower compared to the other AQM schemes (see Figure 5.10).

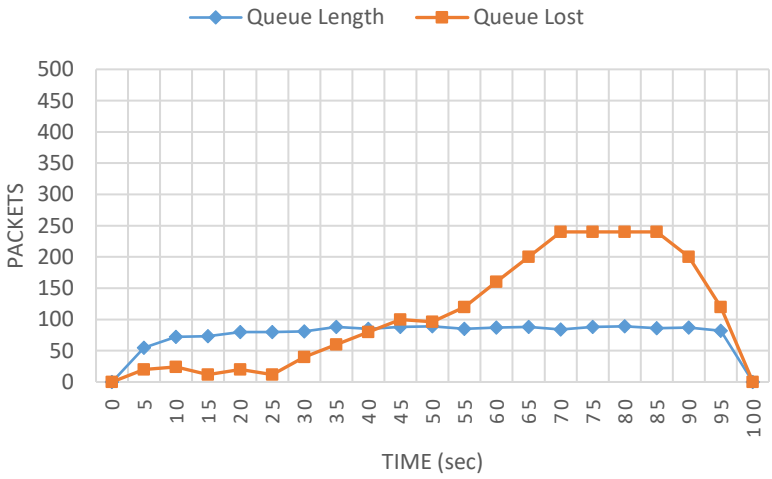

Figure 5.10. Queue Length and Queue Loss for HFAQM

From all of the above, it can be concluded that HFAQM achieved notable stability compared with the other AQM schemes. Moreover, HFAQM increased the queue loss

gradually as the number of sources increased with average queue length 82.4 packets and average queue loss 117 packets. Moreover, when the sources started increasing at 25th second until the last increase at 75th second, the average queue loss for HFAQM was 86 packets and average queue length is 132 packets, and the rest of the schemes is listed below in the table 5.1 during same period.

Table 5.1

The average queue length and queue loss for AQM schemes (by packets)

AQM scheme	From 25th to 75th sec.		Overall	
	Queue length	Queue loss	Queue length	Queue loss
RED	53.3	137.8	55.1	129.8
REM	71.4	152.3	65.4	159.05
AVQ	89.5	139.6	87.8	155.1
CoDel	3.3	269.2	3.4	260.2
HFAQM	86	132.3	82.4	117.05

From the table, HFAQM has a large comparative average queue length and approximately low average queue loss in the period when the sources stated increasing in the scenario. Therefore, HFAQM has more adaptive to the network changes than the rest of schemes. Thus, HFAQM is increased the stability in the WLAN by adapting the network rapid changing due to use adaptive parameter $d_l^*(t)$ in CFM probability function. The next section discusses the results presented in Sections 5.3.1 and 5.3.2.

5.4 Results and Discussion

This section discusses and analyses HFAQM's performance in WLAN network environments, based on the experimental results reported in Sections 5.3.1 and 5.3.2. It presents a summary of several performance metrics conducted, to take a closer look into HFAQM behaviour under the mentioned experiments. Specifically, it analyzes the results of each individual metric, namely fairness, queue length, queue loss, jitter,

throughput and outgoing link utilization, to provide a deeper understanding of HFAQM behaviour in WLAN networks environment.

Fairness: is one of the early objectives of AQM schemes, other than avoiding and controlling congestion. However, networks can suffer from unfairness when multiple sources share the same link to transfer their data, especially when these sources use different types of flow (responsive and nonresponsive). Using the Jain Fairness Index, HFAQM was found to achieve a higher fairness level comparing with the other AQM schemes as provided in Figure 5.4. The explanation is that HFAQM has a hybrid CIM mechanism that uses both queue delay and source rate to indicate congestion and measure the fairness level. The hybrid indication mechanism proved its ability to achieve higher fairness than single-parameter indication mechanisms such as RED and CoDel. Using queue delay and source rate for congestion indication gave better results than other combinations such as queue length and sources rate used by REM and AVQ. Besides, the adaptation of CIM to measure fairness also enabled HFAQM to provide fairness in very early stages. Based on Tukey method [138] to find the significance of HFAQM with other schemes by using the Jain Fairness Index value for each scheme. Moreover the result is presented in the table 5.2 below. From the table, it can be notice that HFAQM have achieved 10% improvement than the closest fairness value by AVQ.

Table 5.2

The average fairness of AQM schemes by using Tukey method

AQM scheme	**RED**	**REM**	**AVQ**	**CoDel**	**HFAQM**
Average	66%	72%	75%	42%	85%

Queue Length: monitoring the queue length is important in analyzing the behaviour of AQM schemes in terms of stability. From the results shown in Figures 5.6-5.10, it is clear that HFAQM achieved better stability than the other AQM schemes; its CFM sets the targeted queue delay time $d_l^*(t)$ instead of α in its probability function, which makes HFAQM more adaptable to changing network conditions. However, its noticeable that HFAQM did achieve a higher queue length than other AQM schemes. The explanation is that HFAQM maintaining fairness among different types of flows which requires high queue delay to increase the accuracy in measuring the price, and that leads to high queue length.

Queue Loss: packet loss is an important metric in network performance measurement. Losing data can be very risky and indicates poor performance especially in terms of QoS. Indeed, the main purpose of avoiding congestion is to reduce packet loss in the network which becomes among the objective of AQM schemes. One of the reasons for packet loss is queue loss which occurs as a result of avoiding congestion, controlling congestion or achieving other objectives such as fairness in the case of HFAQM. From Figures 5.6-5.10, it can be seen that HFAQM has the lowest queue loss, comparing with the rest, thanks to its feedback mechanism which uses hybridization between dropping and marking the packets.

Outgoing Link Utilization: the main objective of the optimization approach is to maximize link utilization, as illustrated in the objective function of Equation 4.7. However, it is constrained by the condition that the aggregate source rate should not exceed the link capacity. The design of HFAQM was based on the optimization approach, in order to achieve high link utilization. Figure 5.11 shows the results of testing HFAQM in a very congested network with different types of flow. Even so,

HF-AQM achieved higher link utilization than RED or CoDel because these have been designed based on other designing approaches without considering source rate in their congestion indicator mechanisms. However, HFAQM also achieved higher utilization than REM and AVQ, even though both schemes were based on the optimization approach and considered source rates in their design. To the superiority of HFAQM results from stabilizing the queue length by managing queue loss using its CFM and feedback mechanism, which combine dropping and marking as feedback signals to the sources. Therefore, it is obvious that reducing packet loss will increase the link utilization.

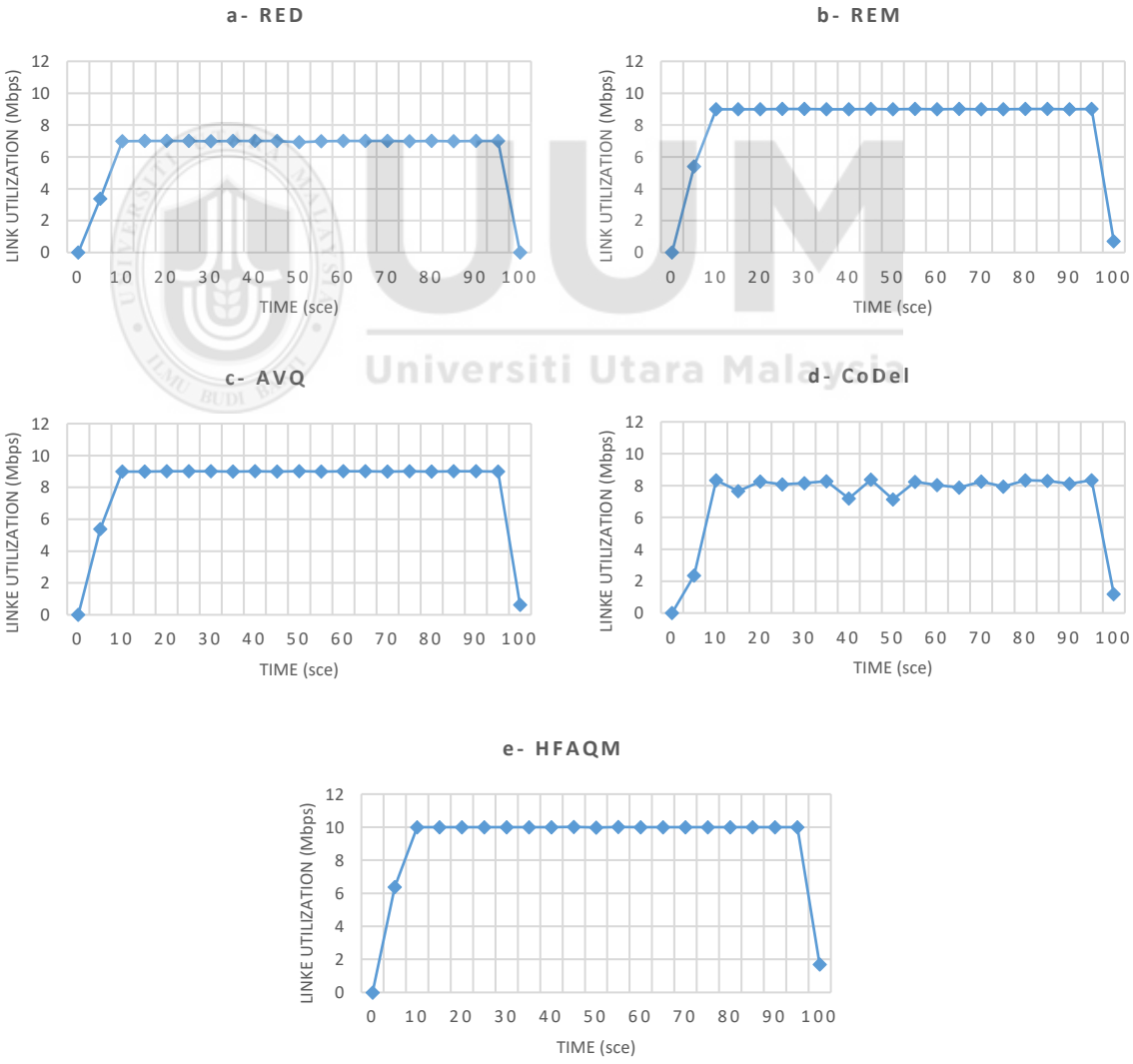

Figure 5.11. Link Utilization for AQM schemes

Throughput: the average throughput was measured using the first scenario shown in Figure 3.9. The results in Figure 5.12 shows that HF-AQM has a higher throughput than the other AQM schemes. CoDel had the lowest value, due to the high level of queue loss. However, HFAQM, AVQ and REM had almost the same performance in terms of throughput, although HFAQM was slightly better because of its lower queue loss.

Figure 5.12. Throughput Result

Jitter: from the test in the first experiment, jitter was induced to measure the effect of queue delay on the delay of the network overall. It can be notice from Figure 5.13 that the network had higher jitter when it used HFAQM as a buffer control scheme. The explanation is that CIM mechanism that been used in HFAQM design tolerates higher queue delay in order to achieve fairness among different types of flow. This queue delay is reflected in the delay of the network overall.

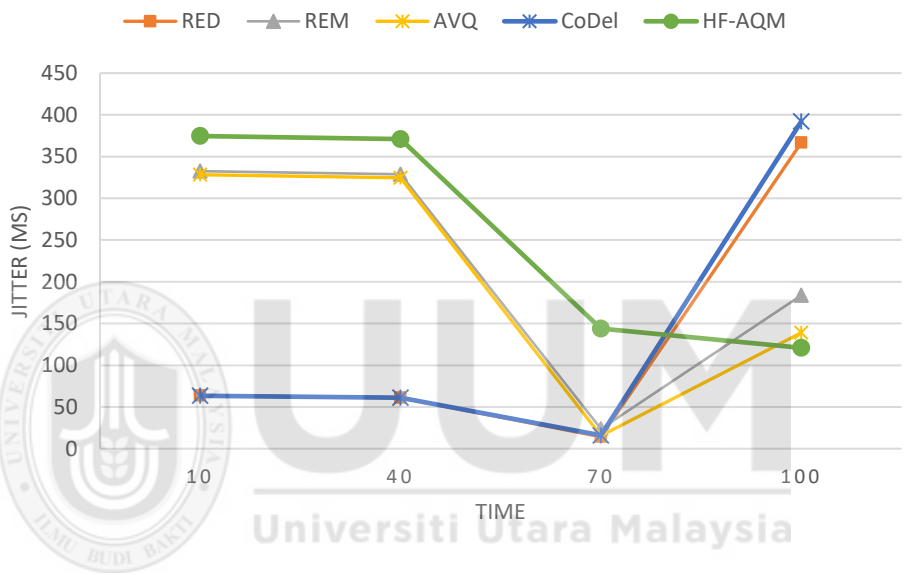

Figure 5.13. Jitter Result

5.5 Summary

This chapter introduced a new AQM named HFAQM, especially for WLAN environment. The design of HFAQM was discussed in this chapter. Then, the proposed scheme was implemented in the NS-2 simulation environment. Following this, the proposed scheme was evaluated inside NS-2 simulator by observing different performance metrics over a variety of scenarios. The evaluation was accomplished by examining HFAQM's fairness, queue length, queue loss, outgoing link utilization, throughput and jitter as compared with RED, REM, AVQ and CoDel schemes in

different scenarios. Furthermore, HFAQM has shown better results than other schemes and more suitable for WLAN environment.

CHAPTER SIX
CONCLUSION AND FUTURE WORK

This thesis aimed at developing a new Active Queue Management (AQM) algorithm called Hybrid Fair AQM (HFAQM) for WLAN networks, and evaluated HFAQM's performance using NS-2 as a simulation tool. Whereas, this chapter concludes this thesis by providing a summary of the research in Section 6.1, including the importance of HFAQM, its findings and discussions. Research contributions are highlighted in Section 6.2. As well as, the limitations of this research are discussed in Section 6.3, with suggestions for future work are mentioned in Section 6.4.

6.1 Summary of the Research

The Internet has grown rapidly over the past years, with a growing demand for algorithms and protocols to improve the performance of the network. Active queue management is one area of critical importance for the operation of networks, for its reactiveness in avoiding and controlling congestion. The main objectives of AQM are detecting and avoiding congestion with low queuing delay and high link utilization; AQM should also achieve other goals: stability, fairness, scalability, robustness and responsiveness.

The earliest AQM algorithms were designed to overcome issues in wired networks such as congestion avoidance and control, which still an open issue. However, the growth of wireless technology raised further issues besides the old ones such as the heterogeneity of different network technologies with their different characteristics and properties. When implementing AQM in wireless networks several new issues should be considered, such as interference, collisions, multipath-fading, propagation distance,

shadowing effects and route failure, and whether the wireless network is of the infrastructured or non-infrastructured type. Therefore, the need for an AQM algorithm to enable wireless networks to perform as well and efficiently as wired networks is urgent.

As explained in Chapter One, the main motivation for this research work was the need for new AQM to improve fairness and stability between different types of flow in infrastructured wireless networks. The study therefore introduced a new HFAQM algorithm to detect and avoid congestion. Furthermore, the proposed algorithm is also capable of providing fairness among responsive and nonresponsive flows with more stability in infrastructured wireless networks, and maximum link utilization with multiple different flows in a fair and stable way with low queuing delay in wireless routers.

The research adapted the Design Research Methodology (DRM) to produce the main stages according to the research objectives. Furthermore, Simulation was chosen as a performance evaluation technique, and was used successfully for implementing, analyzing and applying HFAQM in a wireless network environment.

Chapter Four introduced the congestion indicator mechanism (CIM), which can be considered as the first indicator to hybridize transmission rate with queue delay to formulate a new congestion indicator mathematically. This hybridization improves the congestion detection accuracy, as shown in the simulation, with validation described in Chapter Three and implementation in Chapter Four.

The HFAQM control function mechanism (CFM) was designed using a new formulated probability equation that worked perfectly with CIM and was implemented

in Chapter Four. This probability equation was derived and formulated according to parameters used in price equations for CIM. CFM was implemented in the C++ programing language and validated using the simulation model.

Two different scenarios were used to study and evaluate the performance of HFAQM. Four different AQM algorithms, named as RED, REM, AVQ and CoDel, were chosen to compare the performance of HFAQM. The first scenario was implemented to test the effectiveness of CIM and CFM in avoiding congestion and recovering the network from the overflow stage, and to test its handling of fairness. Whereas, the second scenario was designed to evaluate stability in the HFAQM algorithm. The results from a series of experimental studies demonstrate that HFAQM improves fairness and stability over the other AQM algorithms, with a noticeable difference in throughput, high link utilization and low queue loss. This result emphasizes the fact that the research objectives have been achieved.

6.2 Research Contributions

The main contribution of this research was to design a new hybrid fair AQM algorithm (HFAQM) that can improve fairness and add stability to wireless network environments. The specific contributions are as follows:

i. The development of a mathematical model that hybridize between queue delay and source rate of a congestion indicator mechanism improved accuracy in avoiding congestion and enhanced fairness in WLAN networks.

ii. The design of the adaptive control function mechanism with enhanced congestion control among different types of flow besides the adaptation to the rapid changing environment improved the stability in the WLAN environment.

iii. The mixed feedback mechanism was designed to increase the aggressiveness of the scheme overall, besides enhancing the stability of the network with increasing throughput and link utilization.

6.3 Research Limitation

Although this study was conducted under careful selection and application of a set of supporting methods and guidelines, it is limited to specific and precise utilization. First, the proposed network topologies used in the implementation are widely used and proven in infrastructured wireless networks. The research was conducted in specific network conditions and environments, and not including all the topologies and structures. Moreover, the number of nodes used in the validation and performance evaluation was limited and fixed, whereas the real networks is unpredictable and changeable. Furthermore, not all types of IEEE 802.11 standards were used between the nodes. Moreover, only TCP-Reno was used during the evaluation phase as responsive flows and short flows, and not all applications were discussed due to the time limitation and experiment sophistication.

6.4 Future Works

The proposed mechanisms in this research improve the performance of HFAQM in a variety of scenarios. However, there are some limitations as well as pending works

that can be pursued for future directions. This section outlines some possible extensions, which are as follows:

Extending congestion indicator mathematical model:

The proposed model in Chapter 4 was designed by using two important parameters; transmission rate and queue delay. Considering a third parameter such as queue length may enhance the model accuracy.

Extending control function mechanism:

The HFAQM control function mechanism has been designed to improve fairness and maximize link utilization by matching the queue delay for all different types of flows. This mechanism has improved fairness to a certain level but optimality may never achieved. Using a pre-flow information mechanism besides HFAQM mechanism may enhance fairness among different types of flows.

Designing an evaluation Model:

Yet there is no specific model to evaluate fairness for different types of AQM algorithms to achieve different objectives. However, as future work we advise to design a flexible performance evaluation framework for evaluating the new AQM algorithms and compared with the past ones fairly and efficiently.

Evaluation of HFAQM in a testbed:

Although HFAQM was evaluated comprehensively and extensively through a validated simulator, its implementation in a real testbed is definitely of interest.

Furthermore, to evaluate HFAQM using real traffic is surely a good extension to be performed in extending the life of this research.

REFERENCES

[1] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion Control Protocols in Wireless Sensor Networks: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 3, pp. 1369–1390, 2014.

[2] C. Xu, J. Zhao, and G.-M. Muntean, "Congestion control design for multipath transport protocols: a survey," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 4, pp. 2948–2969, 2016.

[3] R. Mehra and M. Saluja, "Adaptive Congestion Control Mechanisms in Mobile Ad-Hoc Networks," *Int. J. Eng. Dev. Res.*, vol. 5, no. 1, pp. 553–559, 2017.

[4] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Congestion Control for Web Real-Time Communication," *IEEE/ACM Trans. Netw.*, vol. PP, no. 99, pp. 1–14, 2017.

[5] J. F. Kurose and K. W. Ross, *Computer Networking a Top-Down Approach*. PEARSON, 2012.

[6] P. Marbach, "Distributed Scheduling and Active Queue Management in Wireless Networks," in *26th IEEE International Conference on Computer Communications (NFOCOM)*, 2007, pp. 2321–2325.

[7] R. Pletka, M. Waldvogel, and S. Mannal, "PURPLE: Predictive active queue management utilizing congestion information," in *Proceedings. 28th Annual IEEE International Conference on Local Computer Networks, 2003. LCN '03.*, 2003, pp. 21–30.

[8] S. S. Kunniyur and R. Srikant, "An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 286–299, Apr. 2004.

[9] R. Jain, "Congestion control in computer networks: Issues and trends," *IEEE Netw.*, vol. 4, no. 3, pp. 24–30, 1990.

[10] L. Hu and A. D. Kshemkalyani, "HRED: a simple and efficient active queue management algorithm," in *Proceedings. 13th International Conference on Computer Communications and Networks, 2004. ICCCN 2004.*, 2004, pp. 387–393.

[11] M. Arpaci and J. Copeland, "An adaptive queue management method for congestion avoidance in TCP/IP networks," in *Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE*, 2000, pp. 309–315.

[12] S. Ryu, C. Rump, and C. Qiao, "Advances in internet congestion control," *Commun. Surv. Tutorials, IEEE*, vol. 5, no. 1, pp. 28–39, 2003.

[13] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM Computer Communication Review*, 1988, no. 60, pp. 1–25.

[14] S. Floyd, R. Gummadi, and S. Shenker, “Adaptive RED: An algorithm for increasing the robustness of RED’s active queue management,” *IOSR J. Comput. Eng.*, vol. 2, no. 6, pp. 20–24, 2001.

[15] R. Seungwan, R. Christopher, and Q. Chunming, “Advances in Active Queue Management (AQM) Based,” *Telecommun. Syst.*, vol. 25, no. 3–4, pp. 317–351, 2004.

[16] W. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, “The BLUE active queue management algorithms,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 513–528, 2002.

[17] E.-C. Park, H. Lim, K.-J. Park, and C.-H. Choi, “Analysis and design of the virtual rate control algorithm for stabilizing queues in TCP networks,” *Comput. Networks*, vol. 44, no. 1, pp. 17–41, Jan. 2004.

[18] C. Long, B. Zhao, X. Guan, and J. Yang, “The Yellow active queue management algorithm,” *Comput. Networks*, vol. 47, no. 4, pp. 525–550, Mar. 2005.

[19] H. Xu, T. A. Wj, X. Zhou, and Y. Zhu, “Ip network control and aqm,” in *Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai*, 2004, no. August, pp. 26–29.

[20] T. Wu, H. Xu, and S. Tian, “End to end congestion control and active queue management,” in *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, 2003, no. November, pp. 2–5.

[21] J. Koo, B. Song, K. Chung, H. Lee, and H. Kahng, “MRED: a new approach to random early detection,” *Proc. 15th Int. Conf. Inf. Netw.*, pp. 347–352, 2001.

[22] R. Pan, B. Prabhakar, and K. Psounis, “CHOKe-a stateless active queue management scheme for approximating fair bandwidth allocation,” in *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2000. IEEE*, 2000, vol. 2, pp. 942–951.

[23] Y. Jiang, M. Hamdi, and J. Liu, “Self adjustable CHOKe: an active queue management algorithm for congestion control and fair bandwidth allocation,” in *Proceedings. Eighth IEEE International Symposium on Computers and Communication, 2003. (ISCC 2003).*, 2003, pp. 1018–1025.

[24] J. Chung and M. Claypool, “Dynamic-CBT--Better Performing Active Queue Management for Multimedia Networking,” in *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 2000.

[25] W. Sun and K. G. Shin, “TCP performance under aggregate fair queueing,” in *IEEE Global Telecommunications Conference, 2004. GLOBECOM ’04.*, 2004, vol. 3, pp. 1308–1313.

[26] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May, "Comparison of tail drop and active queue management performance for bulk-data and web-like internet traffic," in *Proceedings. Sixth IEEE Symposium on Computers and Communications, 2001.*, 2001, pp. 122–129.

[27] G. de Veciana and M. Borrego, "Minimizing queue variance using randomized deterministic marking," in *IEEE Global Telecommunications Conference GLOBECOM'01. (Cat. No.01CH37270)*, 2001, vol. 4, pp. 2368–2372.

[28] M. K. Agarwal, R. Gupta, and V. Kargaonkar, "Link utilization based AQM and its performance," in *IEEE Global Telecommunications Conference, GLOBECOM '04. 2004.*, 2004, vol. 2, pp. 713–718.

[29] B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choudhury, "Design considerations for supporting TCP with per-flow queueing," in *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1998, vol. 1, pp. 299–306.

[30] D. Lin and R. Morris, "Dynamics of random early detection," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 127–137, Oct. 1997.

[31] K. Nichols and V. Jacobson, "Controlling queue delay," *Commun. ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[32] C. Zhu, O. W. W. Yang, J. Aweya, M. Ouellette, and D. Y. Montuno, "A Comparison of Active Queue Management Algorithms Using the OPNET Modeler," *IEEE Commun. Mag.*, vol. 40, no. 6, pp. 158–167, 2002.

[33] S. S. Oruganti and M. Devetsikiotis, "Analyzing robust active queue management schemes: a comparative study of predictors and controllers," in *IEEE International Conference on Communications, 2003. ICC '03.*, 2003, vol. 3, pp. 1531–1536.

[34] L. Le, J. Aikat, K. Jeffay, and F. D. Smith, "The effects of active queue management on web performance," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '03*, 2003, p. 265.

[35] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "REM: active queue management," *Network, IEEE*, vol. 15, no. 3, pp. 48–53, 2001.

[36] R. Adams, "Active Queue Management: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 3, pp. 1425–1476, 2013.

[37] J. Sun, K. Ko, and G. Chen, "PD-RED: to improve the performance of RED," *IEEE Commun. Lett.*, vol. 7, no. 8, pp. 406–408, 2003.

[38] M. Azizoglu, "On the dropping probability function in active queue management schemes," in *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, 2001, vol. 4, pp.

2511–2516.

[39] X. Deng, S. Yi, G. Kesidis, and C. R. Das, “Stabilized virtual buffer (SVB) - an active queue management scheme for Internet quality-of-service,” in *Global Telecommunications Conference, 2002. GLOBECOM . IEEE*, 2002, vol. 2, pp. 1628–1632.

[40] G. Thiruchelvi and J. Raja, “A survey on active queue management mechanisms,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 8, no. 12, pp. 130–145, 2008.

[41] F. Yanfie, R. Fengyuan, and L. Chuang, “Design a PID controller for active queue management,” in *Proceedings. Eighth IEEE International Symposium on Computers and Communication, 2003.(ISCC 2003).*, 2003, pp. 985–990.

[42] N. Cardwell, S. Savage, and T. Anderson, “Modeling the performance of short TCP connections,” *Techical Rep.*, 1998.

[43] S. H. Low, S. Member, and D. E. Lapsley, “Optimization Flow Control, I: Basic Algorithm and Convergence,” *IEEE/ACM Trans. Networking,* vol. 7, no. 6, pp. 1–16, 1999.

[44] W. Wu, Y. Ren, and X. Shan, “Stability analysis on active queue management algorithms in routers,” in *Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems MASCOTS 2001,* 2001, pp. 125–132.

[45] B. Braden *et al.*, “Recommendations on queue management and congestion avoidance in the Internet,” 1998.

[46] F. Baker and G. Fairhurst, “IETF Recommendations Regarding Active Queue Management,” 2015.

[47] C. Lai, K.-C. Leung, and V. O. K. Li, “Enhancing AQM to combat wireless losses,” *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 6, pp. 1630–1638, Jun. 2011.

[48] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[49] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-Host Congestion Control for TCP,” *IEEE Commun. Surv. Tutorials*, vol. 12, no. 3, pp. 304–342, 2010.

[50] P. Marbach and Y. Lu, “Active Queue Management and Scheduling for Wireless Networks: The Single-Cell Case,” in *40th Annual Conference on Information Sciences and Systems 2006*, 2006, pp. 1560–1565.

[51] K. Chavan, R. G. Kumar, M. N. Belur, and A. Karandikar, “Robust Active Queue Management for Wireless Networks,” *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 6, pp. 1630–1638, Nov. 2011.

[52] F. Kelly, A. Maulloo, and D. Tan, “Rate control for communication networks: shadow prices, proportional fairness and stability,” *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Apr. 1998.

[53] S. Ryu and C. Cho, “PI-PD-controller for robust and adaptive queue management for supporting TCP congestion control,” in *Simulation Symposium, 2004. Proceedings. 37th Annual*, 2004, pp. 132–139.

[54] F. Ren, C. Lin, and B. Wei, “Design a robust controller for active queue management in large delay networks,” in *Proceedings. ISCC 2004, Ninth International Symposium on Computers and Communications, 2004.*, 2004, vol. 2, pp. 748–754.

[55] D. Bauso, L. Giarré, and G. Neglia, “About the stability of active queue management mechanisms,” in *Proceedings of the American Control Conference.*, 2004, vol. 4, pp. 2954–2959.

[56] H. Han, C. V Hollot, Y. Chait, and V. Misra, “TCP networks stabilized by buffer-based AQMs,” in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004, vol. 2, pp. 964–974.

[57] S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 458–472, 1999.

[58] M. Nabeshima, “Improving the performance of active buffer management with per-flow information,” *IEEE Commun. Lett.*, vol. 6, no. 7, pp. 306–308, 2002.

[59] A. Kamra *et al.*, “SFED: a rate control based active queue management discipline,” in *IBM India Research Laboratory Research Report*, 2000.

[60] R. Pletka, A. Kind, M. Waldvogel, and S. Mannal, “Closed-loop congestion control for mixed responsive and non-responsive traffic,” in *IEEE Global Telecommunications Conference. GLOBECOM’03.*, 2003, vol. 7, pp. 4180–4185.

[61] M. Claypool, R. Kinicki, and M. Hartling, “Active queue management for web traffic,” in *IEEE International Conference on Performance, Computing, and Communications,* 2004, pp. 531–538.

[62] W. Feng, A. Kapadia, and S. Thulasidasan, “GREEN: proactive queue management over a best-effort network,” in *IEEE Global Telecommunications Conference, GLOBECOM’02.*, 2002, vol. 2, pp. 1774–1778.

[63] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, “Stochastic fair blue: A queue management algorithm for enforcing fairness,” in *Proceedings. IEEE Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2001*, 2001, vol. 3, pp. 1520–1529.

[64] G. Chatranon, M. A. Labrador, and S. Banerjee, “BLACK: detection and preferential dropping of high bandwidth unresponsive flows,” in *IEEE*

International Conference on Communications, ICC'03., 2003, vol. 1, pp. 664–668.

[65] A. Lakshmikantha, C. L. Beck, and R. Srikant, "Robustness of real and virtual queue-based active queue management schemes," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 81–93, 2005.

[66] P. Mrozowski and A. Chydzinski, "On the stability of AQM algorithms," in *Proc. of Applied Computing Conference*, 2009, pp. 276–281.

[67] F. Ren, C. Lin, and B. Wei, "A robust active queue management algorithm in large delay networks," *Comput. Commun.*, vol. 28, no. 5, pp. 485–493, 2005.

[68] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Trans. Networking,* vol. 11, no. 4, pp. 525–536, 2003.

[69] F. Kelly, "Charging and rate control for elastic traffic," *Eur. Trans. Telecommun.*, vol. 8, no. 1, pp. 33–37, 1997.

[70] F. Kelly, "Mathematical modelling of the Internet," in *The Fourth International Congress on Industrial and Applied Mathematics*, 1999, vol. 33, no. July, pp. 1–20.

[71] R. Zhu, H. Teng, and J. Fu, "A predictive PID controller for AQM router supporting TCP with ECN," in *Proceedings. The 2004 Joint Conference of the 10th Asia-Pacific Conference on Communications, 2004 and the 5th International Symposium on Multi-Dimensional Mobile Communications*, 2003, vol. 1, pp. 356–360.

[72] T. Ziegler, "On averaging for active queue management congestion avoidance," in *IEEE Symposium on Computers and Communications (ISCC)*, 2002, p. 867.

[73] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *ACM SIGCOMM Computer Communication Review*, 2000, vol. 30, no. 4, pp. 151–160.

[74] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *Proceedings. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2001.*, 2001, vol. 3, pp. 1510–1519.

[75] M. Kisimoto, H. Ohsaki, and M. Murata, "On transient behavior analysis of random early detection gateway using a control theoretic approach," in *Proceedings of the 2002 International Conference on Control Applications,* 2002, vol. 2, pp. 1144–1146.

[76] R. Cartas, J. Orozco, J. Incera, and D. Ros, "A fairness study of the adaptive RIO active queue management algorithm," in *Proceedings of the Fifth Mexican International Conference in Computer Science, ENC 2004.*, 2004, pp. 57–63.

[77] J. Chen, C. Hu, and Z. Ji, "Self-tuning random early detection algorithm to improve performance of network transmission," *Math. Probl. Eng.*, vol. 2011, 2010.

[78] W. Feng, "Improving Internet congestion control and queue management algorithms," IBM Research, 1999.

[79] G. F. Ahammed and R. Banu, "Analyzing the performance of Active Queue Management Algorithms," *Int. J. Comput. Networks Commun.*, vol. 2, no. 2, pp. 1–19, 2010.

[80] H. Chandra, A. Agarwal, and T. Velmurugan, "Analysis of active queue management algorithms & their implementation for TCP/IP networks using OPNET simulation tool," *Int. J. Comput. Appl.*, vol. 6, no. 11, 2010.

[81] M. Kwon and S. Fahmy, "Comparison of load-based and queue-based active queue management algorithms," in *ITCom 2002: The Convergence of Information Technologies and Communications*, 2002, pp. 35–46.

[82] B. Zheng and M. Atiquzzaman, "DSRED: improving performance of active queue management over heterogeneous networks," in *IEEE International Conference on Communications, 2001. ICC 2001.*, 2001, vol. 8, pp. 2375–2379.

[83] C. Wang, B. Li, Y. Thomas Hou, K. Sohraby, and Y. Lin, "LRED: a robust active queue management scheme based on packet loss ratio," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004, vol. 1.

[84] T. Alemu and A. Jean-Marie, "Dynamic configuration of RED parameters," in *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, 2004, vol. 3, pp. 1600–1604.

[85] S. De Cnodder, K. Pauwels, and O. Elloumi, "A rate based RED mechanism," in *The 10th International Workshop on Network and Operating System Support for Digital Audio and Video, Chapel Hill, North Carolina*, 2000, pp. 1–9.

[86] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2000, vol. 3, pp. 1435–1444.

[87] T. Eguchi, H. Ohsaki, and M. Murata, "On control parameters tuning for active queue management mechanisms using multivariate analysis," in *Symposium on Applications and the Internet, 2003. Proceedings. 2003*, 2003, pp. 120–127.

[88] Y. Chait, C. V Hollot, V. Misra, S. Oldak, D. Towsley, and W.-B. Gong, "Fixed and adaptive model-based controllers for active queue management," in *Proceedings of the 2001 American Control Conference, 2001.*, 2001, vol. 4, pp. 2981–2986.

[89] F. M. Anjum and L. Tassiulas, “Fair bandwidth sharing among adaptive and non-adaptive flows in the Internet,” in *INFOCOM’99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1999, vol. 3, pp. 1412–1420.

[90] E. Bowen, C. Jeffries, L. Kencl, A. Kind, and R. Pletka, “Bandwidth allocation for non-responsive flows with active queue management,” in *2002 International Zurich Seminar on Broadband Communications, 2002. Access, Transmission, Networking.*, 2002, pp. 11–13.

[91] G. Chatranon, M. Labrador, and S. Banerjee, “A survey of TCP-friendly router-based AQM schemes,” *Comput. Commun.*, vol. 27, no. 15, pp. 1424–1440, Sep. 2004.

[92] G. Chatranon, M. Labrador, and S. Banerjee, “Fairness of AQM schemes for TCP-friendly traffic,” in *IEEE Global Telecommunications Conference, 2004. GLOBECOM ’04.*, 2004, vol. 2, pp. 725–731.

[93] L. Massoulié and J. Roberts, “Bandwidth sharing: objectives and algorithms,” in *INFOCOM’99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1999, vol. 3, pp. 1395–1403.

[94] S. Wen, Y. Fang, and H. Sun, “Differentiated bandwidth allocation with TCP protection in core routers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 1, pp. 34–47, 2009.

[95] S. S. Kunniyur and R. Srikant, “Stable, scalable, fair congestion control and AQM schemes that achieve high utilization in the internet,” *IEEE Trans. Autom. Control.*, vol. 48, no. 11, pp. 2024–2028, 2003.

[96] S. Kunniyur and R. Srikant, “End-to-end congestion control schemes: Utility functions, random losses and ECN marks,” *IEEE/ACM Trans. Networking,* vol. 11, no. 5, pp. 689–702, 2003.

[97] G. Feng, A. K. Agarwal, A. Jayaraman, and C. K. Siew, “Modified RED gateways under bursty traffic,” *IEEE Commun. Lett.*, vol. 8, no. 5, pp. 323–325, 2004.

[98] Y. Hong and O. W. W. Yang, “Design of TCP traffic controllers for AQM routers based on phase margin specification,” in *Workshop on High Performance Switching and Routing, HPSR.*, 2004, pp. 314–318.

[99] T. Jain, M. P. Tahiliani, and others, “Performance Evaluation of CoDel for Active Queue Management in Wired-Cum-Wireless Networks,” in *2014 Fourth International Conference on Advanced Computing & Communication Technologies (ACCT),* 2014, pp. 381–385.

[100] D. M. Raghuvanshi, B. Annappa, and M. P. Tahiliani, “On the effectiveness of CoDel for active queue management,” in *2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT),* 2013, pp. 107–114.

[101] T. Sharma, “Controlling Queue Delay (CoDel) to counter the Bufferbloat Problem in Internet,” *Int. J. Curr. Eng. Technol.*, vol. 4, no. 3, pp. 2210–2215, 2014.

[102] N. Khademi, D. Ros, and M. Welzl, “The New AQM Kids on the Block : Much Ado About Nothing ?,” *Tech. Rep. 434*, pp. 1–24, 2013.

[103] P. Offermann, O. Levina, M. Schönherr, and U. Bub, “Outline of a design science research process,” in *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, 2009, p. 7.

[104] L. Blessing and A. Chakrabarti, *DRM, a design research methodology*. Springer Science, 2009.

[105] O. Balci, “Verification validation and accreditation of simulation models,” in *Proceedings of the 29th conference on Winter simulation*, 1997, pp. 135–141.

[106] R. G. Sargent, “Verification and validation of simulation models,” in *Proceedings of the 37th conference on Winter simulation*, 2005, pp. 130–143.

[107] O. Balci, “Verification, validation, and testing,” *Handb. Simul.*, vol. 10, pp. 335–393, 1998.

[108] R. B. Whitner and O. Balci, “Guidelines for selecting and using simulation model verification techniques,” in *Proceedings of the 21st conference on Winter simulation*, 1989, pp. 559–568.

[109] L. G. Birta and F. N. Özmizrak, “A knowledge-based approach for the validation of simulation models: the foundation,” *ACM Trans. Model. Comput. Simul.*, vol. 6, no. 1, pp. 76–98, 1996.

[110] R. S. Pressman, *Software engineering: a practitioner’s approach*. Palgrave Macmillan, 2005.

[111] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*. John Wiley & Sons, 2011.

[112] A. Habbal, “TCP Sintok: Transmission control protocol with delay-based loss detection and contention avoidance mechanisms for mobile ad hoc networks,” Universiti Utara Malaysia, 2014.

[113] S. H. Trivedi, “Software Testing Techniques,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 2, pp. 433–439, 2012.

[114] K. J. Cohen and R. M. Cyert, “Computer models in dynamic economics,” *Q. J. Econ.*, vol. 75, no. 1, pp. 112–127, 1961.

[115] J. W. Forrester, “Industrial dynamics,” *J. Oper. Res. Soc.*, vol. 48, no. 10, pp. 1037–1041, 1997.

[116] R. L. Van Horn, “Validation of simulation results,” *Manage. Sci.*, vol. 17, no.

5, pp. 247–258, 1971.

[117] C. F. Hermann, “Validation problems in games and simulations with special reference to models of international politics,” *Syst. Res. Behav. Sci.*, vol. 12, no. 3, pp. 216–231, 1967.

[118] R. D. Wright, “Validating dynamic models: An evaluation of tests of predictive power,” in *Proceedings of 1972 Summer Computer Simulation Conference*, 1972, pp. 13–16.

[119] D. K. Miller, “Validation of computer simulations in the social sciences,” in *Proceedings of the Sixth Annual Conference on Modeling and Simulation*, 1975, pp. 743–746.

[120] I. Sommerville, “Software Engineering. International computer science series,” *ed Addison Wesley*, 2004.

[121] J. Mo and J. Walrand, *Performance Modeling of Communication Networks*, 1st ed. Morgan & Claypool Publishers, 2009.

[122] A. M. M. Habbal, “Tcp Sintok: Transmission Control Protocol with Dela-based Loss Detection and contention Avoidance Mechanisms for Mobile Ad Hoc Networks,” College of Arts and Sciences, Universiti Utara Malaysia, 2014.

[123] A. C. M. Suki, “Slight-Delay Shaped Variable Bit Rate (SD-SVBR) Technique for Video Transmission,” College of Arts and Sciences, Universiti Utara Malaysia, 2011.

[124] R. Jain, *The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling*. New York: John Willey, 1991.

[125] G. Osman, “Scaleable and Smooth TCP-Friendly Receiver-Based Layered Multicast Protocol,” Universiti Utara Malaysia, 2008.

[126] D. Thomas, A. Joiner, W. Lin, M. Lowry, and T. Pressburger, “The unique aspects of simulation verification and validation,” in *Aerospace Conference, 2010 IEEE*, 2010, pp. 1–7.

[127] B. L. Ong, “A hybrid mechanism for SIP over IPv6 macromobility and micromobility management protocols,” Universiti Utara Malaysia, 2008.

[128] O. M. D. Al-Momani, “Dynamic redundancy forward error correction mechanism for the enhancement of Internet-based video streaming,” Universiti Utara Malaysia, 2010.

[129] O. M. Hasbullah, “An Innovative Signal Detection Algorithm in Facilitating the Cognitive Radio Functionality for Wireless Regional Area Network Using Singular Value Decomposition,” College of Arts and Sciences, Universiti Utara Malaysia, 2011.

[130] M. Köksal, “A survey of network simulators supporting wireless networks,” *Middle East Tech. Univ. Ankara, Turkey*, vol. 20, pp. 1–11, 2008.

[131] G. F. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. J. Reed, “Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed,” *WSEAS Trans. Comput.*, vol. 2, no. 3, pp. 700–707, 2003.

[132] P. P. Garrido, M. P. Malumbres, and C. T. Calafate, “ns-2 vs. OPNET: a comparative study of the IEEE 802.11 e technology on MANET environments,” in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, p. 37.

[133] A. Deepak, K. S. Shravya, and M. P. Tahiliani, “Design and Implementation of AQM Evaluation Suite for ns-3,” *Work. ns-3 - WNS3 2017*, no. 2, pp. 87–94, 2017.

[134] A. Marin, S. Rossi, A. Bujari, and C. Palazzi, “Performance evaluation of AQM techniques with heterogeneous traffic,” in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2016, pp. 194–199.

[135] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984.

[136] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, vol. 23. Prentice hall Englewood Cliffs, NJ, 1989.

[137] D. G. Luenberger, Y. Ye, and others, *Linear and nonlinear programming*, vol. 2. Springer, 1984.

[138] R. Lowry, “One way anova-independent samples,” *Vassar. edu*, 2008.